To be submitted to ApJS

GENASIS: *GEN*ERAL *A*STROPHYSICAL *SI*MULATION *S*YSTEM. II. NONRELATIVISTIC HYDRODYNAMICS

Christian Y. Cardall[1,2], Reuben D. Budiardja[1,2,3,4], Eirik Endeve[5], and Anthony Mezzacappa[1,2,5]

cardallcy@ornl.gov

ABSTRACT

In this paper, the second in a series, we document the algorithms and solvers for compressible nonrelativistic hydrodynamics implemented in GenASiS (*Gen*eral *A*strophysical *Si*mulation *S*ystem)—a new code being developed initially and primarily, though by no means exclusively, for the simulation of core-collapse supernovae. In the `Mathematics` division of GenASiS we introduce `Solvers`, which includes finite-volume updates for generic hyperbolic `BalanceEquations` and ordinary differential equation integration `Steps`. We also introduce the `Physics` division of GenASiS; this extends the `Manifolds` division of `Mathematics` into physical `Spaces`, defines `StressEnergies`, and combines these into `Universes`. We benchmark the hydrodynamics capabilities of GenASiS against many standard test problems; the results illustrate the basic competence of our implementation, demonstrate the manifest superiority of the HLLC over the HLL Riemann solver in a number of interesting cases, and provide preliminary indications of the code's ability to scale and to function with cell-by-cell fixed-mesh refinement.

Subject headings: hydrodynamics – methods: numerical

1. INTRODUCTION

The elucidation of the explosion mechanism of core-collapse supernovae is one example of a multiphysics and multiscale astrophysics problem that presses against the boundaries of available supercomputer software and hardware. After core collapse and the launch of the supernova shock wave, the subsequent evolution—eventually leading to the disruption of a star with mass greater than $\sim 8\,M_\odot$—likely involves a rich array of fluid phenomena, including strong shocks, rotation, fluid instabilities, and turbulent cascades (in addition to neutrino radiation transport, self-gravity, and nuclear equations of state and reaction kinetics; e.g. Mezzacappa 2005; Woosley and Janka 2005; Kotake et al. 2006; Janka et al. 2007; Janka 2012).

GenASiS (*Gen*eral *A*strophysical *Si*mulation *S*ystem) is a new code being developed initially and primarily, though by no means exclusively, for the simulation of core-collapse supernovae on the world's leading capability supercomputers. Its design facilitates reference to multiple algorithms, solvers, and physics and numerics choices with

[1]Physics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6354, USA

[2]Department of Physics and Astronomy, University of Tennessee, Knoxville, TN 37996-1200, USA

[3]Joint Institute for Heavy Ion Research, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6374, USA

[4]National Institute for Computational Sciences, University of Tennessee, Knoxville, TN 37996, USA

[5]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6354, USA

the same abstracted names and/or interfaces. In GenASiS this is accomplished through use of Fortran 2003 features that support the object-oriented programming paradigm (e.g. Reid 2007; Adams et al. 2008). n addition to facilitating flexibility in terms of physics and numerics choices, the principles of object-oriented design may also greatly facilitate algorithms associated with adaptive mesh refinement (AMR). The rationale behind GenASiS—including discussions and examples of object-oriented design, and our choice to develop the code with an eye towards cell-by-cell AMR (e.g. Khokhlov 1998; Teyssier 2002; Gittings et al. 2008)—has been further explained in a separate paper (Cardall et al. 2012, hereafter referred to as Paper I).

In this paper—the second in a series—we describe the initial capabilities of GenASiS for compressible nonrelativistic hydrodynamics, including an example with a fixed multilevel grid of a type suitable for gravitational collapse. The equations of hydrodynamics can be categorized as a system of hyperbolic balance equations, for which a vast mathematical literature on different solvers exists (e.g. Shu 1997; LeVeque 2002; Toro 2009, and references therein). The multiphysics nature of core-collapse supernova explosion mechanism simulations requires that the hydrodynamics solver accommodate a non-ideal equation of state and other physical ingredients (e.g. magnetic fields, gravity, neutrino-matter interactions, etc.) in a robust manner. Core-collapse supernovae are ultimately general relativistic systems, and the hydrodynamics solvers should be generalizable to a relativistic description. Moreover, the solver must be able to accurately describe the formation and evolution of shocks and other discontinuities. Our choice to use AMR to at least partially deal with the multiscale nature of core-collapse supernovae also puts constraints on the choice of hydrodynamics solver.

Finite volume methods based on approximate Riemann solvers are good candidates under these various considerations. The hydrodynamics solvers in GenASiS are second-order accurate, are based on the integral formulation of the underlying hyperbolic system, and use the method of lines approach to the solution of partial differential equations (e.g. Shu 1997; Kurganov et al. 2001). Second-order spatial accuracy is achieved with monotonic linear spatial interpolation (e.g. Kurganov and Tadmor 2000). We employ the so-called HLL family of Riemann solvers (Harten et al. 1983; Einfeldt 1988; Toro et al. 1994; Kurganov et al. 2001) to compute intercell fluxes. Second-order temporal accuracy is achieved with a Total Variation Diminishing (TVD) Runge-Kutta method (e.g. Shu 1997). In particular, schemes based on HLL-type Riemann solvers rely on minimal information about the eigenstructure of the underlying hyperbolic system, and have been promoted as general black-box solvers for conservation laws and related equations (Kurganov and Tadmor 2000). Indeed, HLL-type Riemann solvers have been designed for a range of hyperbolic equations, including nonrelativistic hydrodynamics and MHD (e.g. Toro et al. 1994; Batten et al. 1997; Linde 2002; Londrillo and Del Zanna 2004; Miyoshi and Kusano 2005), and special *and* general relativistic hydrodynamics and MHD (e.g. Del Zanna and Bucciantini 2002; Del Zanna et al. 2003; Gammie et al. 2003; Duez et al. 2005; Mignone and Bodo 2005; Del Zanna et al. 2007; Mignone et al. 2009). HLL-type Riemann solvers have also been used for general relativistic neutrino radiation-hydrodynamics simulations of core-collapse supernovae (Müller et al. 2010; Müller et al. 2012).

Figure 1, first presented and further explained in Paper I, shows the high-level structure of the core of GenASiS. Both `Modules` and `Programs` contain divisions labeled `Physics`, `Mathematics`, and `Basics`. On the `Modules` side, these categories contain class implementations; dual to these on `Programs` side, in almost complete one-to-one correspondence, are unit tests that exercise the capabilities and provide example usage of the individual classes. `Programs` also has another division—`Applications`—intended for all drivers aimed at any purpose beyond unit tests, ranging from the solution of simple physical test problems to the execution of production-scale multiphysics research simulations.

With the top-down overview afforded by Figure 1 in mind, this paper continues our series on GenASiS by describing its implementation of nonrelativistic hydrodynamics. Paper I covers `Basics` and one of the divisions of `Mathematics`: cell-by-cell refinable `Manifolds`. Another division of `Mathematics` is introduced in this paper:

Solvers includes finite-volume updates for generic hyperbolic BalanceEquations and ordinary differential equation integration Steps. We also introduce the Physics division of GenASiS; this extends the Manifolds division of Mathematics into physical Spaces (in particular a Galilean space), defines StressEnergies (in particular Fluids), and combines these into Universes (in particular a FreeFluidForm whose evolution draws on BalanceEquations). While two sample programs are discussed in Paper I, here we present our first Applications in the form of several standard test problems against which we benchmark the hydrodynamics capabilities of GenASiS.

2. SOLVERS

Solvers is the only division of Mathematics that we report on in this paper. The Mathematics division of GenASiS contains generic mathematical constructs and solvers that are as agnostic as possible with regard to the specifics of any particular system, in order that they might in principle be used for a wide variety of physical (or other) problems. (For the place of Mathematics in the overall scheme of GenASiS, see Figure 1.) Another division of Mathematics shown in the left diagram of Figure 2—Manifolds—is described in Paper I. As illustrated in the middle diagram of Figure 2, Solvers is subdivided into BalanceEquations and Steps. While Steps is a 'leaf' division (containing only classes and no further subdivisions), the right diagram of Figure 2 shows that BalanceEquations includes the leaf divisions BalanceEquationBasics, Reconstructions, RiemannSolvers, and BalanceEquationUpdates.

2.1. Balance Equations

Hyperbolic balance equations are of the general form

$$\frac{\partial \mathcal{U}}{\partial t} + \nabla \cdot \mathcal{F}(\mathcal{U}) = \mathcal{S}(\mathcal{U}). \tag{1}$$

Here $\mathcal{U}$ is a vector of 'conserved' or 'balanced' variables, whose rate of change is determined by the divergence of the fluxes $\mathcal{F}(\mathcal{U})$ and by the sources $\mathcal{S}(\mathcal{U})$. In the absence of source terms, a balance equation reduces to a 'conservation law': for an infinitesimal volume dV, the rate of change of $\mathcal{U}\,dV$ is equal to the integrated flux $-\oint_S \mathcal{F}(\mathcal{U}) \cdot dS$ flowing through the closed surface S surrounding dV. The system described by Equation (1) is said to be hyperbolic if the Jacobian matrix associated with the flux divergence is diagonalizable and has real eigenvalues (e.g. LeVeque 2002). Associated with $\mathcal{U}$ are two other sets of variables: 'primitive' variables $\mathcal{W}$, often equal in number to the number of balanced variables; and some additional 'auxiliary' variables $\mathcal{A}$, typically determined by one or more closure relations involving the primitive variables (e.g. an 'equation of state' in the case of hydrodynamics). The division BalanceEquationsBasics of BalanceEquations (see the right diagram in Figure 2) contains the class BalancedVariableGroupTemplate, which outlines some basic infrastructure associated with a set of variables to be addressed by balance equations. This class is an extension of VariableGroupForm. It is abstract, with deferred methods that include ComputeBalanced, ComputePrimitive, and ComputeAuxiliary, which respectively compute the variable sets $\mathcal{U}$, $\mathcal{W}$, and $\mathcal{A}$ in terms of the others. (See Paper I for more on VariableGroupForm, class inheritance using the Fortran 2003 extends keyword, and abstract classes with deferred methods.)

The divergence structure of hyperbolic balance equations lends itself naturally to a finite-volume approach (e.g. LeVeque 2002). Spatial discretization involves taking the volume average of Equation (1) over each of the cuboid cells

in our multilevel grid structure:

$$\frac{\partial \langle \mathcal{U} \rangle}{\partial t} = -\frac{1}{V} \sum_q \left[\left(A_q \langle \mathcal{F}^q \rangle \right)_{q\rightarrow} - \left(A_q \langle \mathcal{F}^q \rangle \right)_{\leftarrow q} \right] + \langle \mathcal{S} \rangle. \tag{2}$$

The sum is over dimensions. Angle brackets of quantities associated with arrowed subscripts denote an area average over an outer ($\rightarrow$) or inner ($\leftarrow$) face of the cell, and angle brackets without such arrowed subscripts indicate a cell volume average. The cell volume and face areas are V and A_q, respectively. While discretized in space, Equation (2) remains continuous in time. (It is also exact, until numerical approximations to volume-averaged sources and area-averaged fluxes on the faces are taken.) Once fully discretized in space, it is viewed as a system of ordinary differential equations, which then can be discretized in time and integrated using standard explicit techniques (e.g. Runge-Kutta methods). This approach is frequently used to design higher order methods for hyperbolic systems (e.g. Shu 1998). (We only consider second-order methods in the work presented here, however.) The spatial order of accuracy can be different from the temporal order of accuracy, although the overall formal accuracy of the scheme is limited to the lower of the two. This general approach—in which all dimensions but one are discretized, so as to allow application of methods for ordinary differential equations—is called the method of lines. It has also been called a semi-discrete method (e.g. Kurganov and Tadmor 2000).

2.1.1. Reconstructions

Before the face-averaged fluxes in Equation (2) can be computed, variable values on the faces must be obtained. This process, called 'reconstruction,' is handled by the classes in the `Reconstructions` division of `BalanceEquations` (see the right diagram in Figure 2). In GenASiS the reconstruction is done by interpolating the primitive variables (as opposed to the characteristic variables; e.g. Shu 1997; LeVeque 2002). We first obtain approximate values for the primitive variables $\mathcal{W}$ on the cell faces, and then use them to compute the conserved and auxiliary variables $\mathcal{U}$ and $\mathcal{A}$. In the work presented here we take the primitive variables to be represented by a linear expansion within a cell. (For cells of equal size, linear reconstruction results in a second-order spatial scheme.) For some point inside the cell—which we shall call the cell center—the values of the primitive variables are equal to the cell volume averages:

$$\mathcal{W}_{\leftrightarrow} = \langle \mathcal{W} \rangle, \tag{3}$$

where the double-headed arrow ($\leftrightarrow$) denotes evaluation at the cell center. (In Cartesian coordinates—which we use in all the test problems in this paper—the cell center defined in this way coincides with the geometric center, i.e. the point equidistant from the centers of all cell faces.) Denoting the first derivative of $\mathcal{W}$ in the q dimension by $\mathcal{D}_q \left[\mathcal{W}_{\leftrightarrow} \right]$, the values at the inner and outer faces are

$$\mathcal{W}_{\leftarrow q} = \mathcal{W}_{\leftrightarrow} - \mathcal{D}_q \left[\mathcal{W}_{\leftrightarrow} \right] \left(q_{\leftrightarrow} - q_{\leftarrow q} \right), \tag{4}$$

$$\mathcal{W}_{q\rightarrow} = \mathcal{W}_{\leftrightarrow} + \mathcal{D}_q \left[\mathcal{W}_{\leftrightarrow} \right] \left(q_{q\rightarrow} - q_{\leftrightarrow} \right), \tag{5}$$

where the last factors are the coordinate distances between the faces and the cell center. Spurious oscillations near discontinuities can be reduced by using a slope limiter that enforces monotonic reconstruction; we use (e.g. Kurganov and Tadmor 2000)

$$\mathcal{D}_q \left[\mathcal{W}_{\leftrightarrow} \right] = \mathrm{MM} \left[\vartheta \left(\frac{\mathcal{W}_{\leftrightarrow} - \mathcal{W}_{-q\leftrightarrow}}{q_{\leftrightarrow} - q_{-q\leftrightarrow}} \right), \left(\frac{\mathcal{W}_{\leftrightarrow q+} - \mathcal{W}_{-q\leftrightarrow}}{q_{\leftrightarrow q+} - q_{-q\leftrightarrow}} \right), \vartheta \left(\frac{\mathcal{W}_{\leftrightarrow q+} - \mathcal{W}_{\leftrightarrow}}{q_{\leftrightarrow q+} - q_{\leftrightarrow}} \right) \right]. \tag{6}$$

Center values in left and right (previous and next) neighboring cells in the q dimension are labeled by subscripts $-q \leftrightarrow$ and $\leftrightarrow q+$ respectively. The minmod function $\mathrm{MM}[\cdot]$ compares its arguments and chooses the one with the smallest

magnitude. If the arguments do not all have the same sign, the minmod function returns zero. The three arguments in Equation (6) are left, centered, and right slopes, with the left and right slopes multiplied by the slope limiter parameter $\vartheta \in [1, 2]$. Higher values of ϑ promote the centered difference, and are therefore less diffusive and more accurate for smooth flows, but also more prone to oscillations in the presence of discontinuities. The value $\vartheta = 1$ is equivalent to the traditional minmod that selects between only the left and right derivatives; this is generally unacceptably diffusive. We often use $\vartheta = 1.4$.

2.1.2. RiemannSolvers

Because variables on cell faces are reconstructed independently for each cell, the values obtained from the cells on the left and right sides of an interface do not generally match up, and must be resolved to obtain a single value of the flux to be applied to both cells. In the finite-volume approach it is natural to consider the two states on the immediate left and right side of the interface as constituting a 'Riemann problem' consisting of two regions of constant data, separated by a single discontinuity, and governed by a one-dimensional version of Equation (1):

$$\frac{\partial \mathcal{U}}{\partial t} + \frac{\partial \mathcal{F}^q(\mathcal{U})}{\partial q} = \mathcal{S}(\mathcal{U}). \tag{7}$$

(For second-order methods, the extension to the multidimensional case is simple, and is achieved by applying the one-dimensional spatial discretization prescription separately in each dimension.) The initial conditions in the two regions are commonly referred to as the 'left' and 'right' states; as applied to cell interfaces, these are

$$(\mathcal{U}_{\text{L}})_{\leftarrow q} = \mathcal{U}_{-q\rightarrow}, \tag{8}$$

$$(\mathcal{U}_{\text{R}})_{\leftarrow q} = \mathcal{U}_{\leftarrow q} \tag{9}$$

at a cell's inner face, and

$$(\mathcal{U}_{\text{L}})_{q\rightarrow} = \mathcal{U}_{q\rightarrow}, \tag{10}$$

$$(\mathcal{U}_{\text{R}})_{q\rightarrow} = \mathcal{U}_{\leftarrow q+} \tag{11}$$

at a cell's outer face, where the subscripts $-q \rightarrow$ and $\leftarrow q+$ respectively denote outer face values of the previous cell and inner face values of the next cell. As a Riemann problem evolves, several waves propagate away from the initial discontinuity into the left and right states, with velocities given by the eigenvalues λ_q of the Jacobian matrix $\partial \mathcal{F}^q / \partial \mathcal{U}$.

A full solution of the Riemann problem consists of finding the 'characteristics,' or trajectories of the several waves admissible by the underlying hyperbolic equations, and determining the (not necessarily constant) values of the variables in the regions bounded by them. For example, in the case of nonrelativistic hydrodynamics (i.e., the Euler equations of gas dynamics), the solution of the Riemann problem may consist of shock waves, rarefaction waves, and contact discontinuities (cf. Figure 3). In general, the procedure of solving the Riemann problem involves utilizing the full characteristic structure of the underlying hyperbolic equations; i.e., the eigenvalues *and* the (left and right) eigenvectors of the Jacobian matrix (e.g. LeVeque 2002). On the one hand, using the most precise characteristic information about the hyperbolic system under consideration certainly results in highly accurate numerical methods. On the other hand, such elaborate Riemann solvers are often designed with specific (and often simplifying) assumptions about the closure relations involved, and are not easily extended to the more general closure relations appearing in multiphysics applications. Moreover, the complexity of the solution procedure increases with the complexity of the system of hyperbolic equations (e.g. Euler equations of gas dynamics versus the equations of ideal magnetohydrodynamics).

In practical applications, it is often desirable to work with only approximate solutions to the Riemann problem at cell interfaces in order to obtain the face-averaged fluxes needed in Equation (2), and in the work presented here we use

two variants of the HLL-type approximate Riemann solvers (Harten et al. 1983; Einfeldt 1988; Toro et al. 1994). They rely on minimal knowledge of the characteristic structure of the underlying hyperbolic system, but can still result in very robust and accurate numerical schemes. This is a desirable property for schemes that are incorporated into multiphysics simulations of non-ideal gases (e.g. simulations of core-collapse supernovae), or schemes for systems where the full characteristic structure of the underlying equations is unknown or very complicated (e.g. the equations of general relativistic radiation hydrodynamics). HLL-type Riemann solvers have been designed for and applied to a range of hyperbolic systems in one *and* multiple spatial dimensions, including nonrelativistic hydrodynamics and MHD (e.g. Toro et al. 1994; Batten et al. 1997; Linde 2002; Londrillo and Del Zanna 2004; Miyoshi and Kusano 2005), special *and* general relativistic hydrodynamics and MHD (e.g. Del Zanna and Bucciantini 2002; Del Zanna et al. 2003; Gammie et al. 2003; Duez et al. 2005; Mignone and Bodo 2005; Del Zanna et al. 2007; Mignone et al. 2009), and radiation hydrodynamics (e.g. González et al. 2007; Farris et al. 2008; Sekora and Stone 2010). Despite their simplicity, the HLL-type Riemann solvers have proven to be competitive with more elaborate Riemann solvers (e.g. the HLLC Riemann solver of Toro et al. 1994, or the HLLD solver of Miyoshi and Kusano 2005).

HLL-type Riemann solvers determine fluxes from the jump conditions that obtain at the boundaries between the regions separated by the propagating waves. Consider an interval $I = [q_a, q_b]$, separated into two regions of constant data by a discontinuity at $q_d(t) \in I$ that travels with speed $\lambda_d = dq_d/dt$. Integrate Equation (7) over I and separate the integral of the time derivative term into its two regions:

$$\frac{\partial}{\partial t}\left[\int_{q_a}^{q_d(t)} \mathcal{U}\, dq + \int_{q_d(t)}^{q_b} \mathcal{U}\, dq\right] = \int_{q_a}^{q_b}\left[-\frac{\partial \mathcal{F}^q(\mathcal{U})}{\partial q} + \mathcal{S}(\mathcal{U})\right] dq. \tag{12}$$

Assuming continuous sources $\mathcal{S}(\mathcal{U})$ (except perhaps at the discontinuity at q_d), the only terms that survive the limit $q_a \to q_d(t)$ from the left and $q_b \to q_d(t)$ from the right are those arising from the time derivatives of the integral endpoints $q_d(t)$ and the trivial integral of the flux derivative:

$$\lambda_d\left(\mathcal{U}_- - \mathcal{U}_+\right) = \mathcal{F}^q\left(\mathcal{U}_-\right) - \mathcal{F}^q\left(\mathcal{U}_+\right), \tag{13}$$

where $\mathcal{U}_-$ and $\mathcal{U}_+$ are the states immediately to the left and right of the discontinuity. These are the jump conditions.

The first variant we use, which we simply label 'HLL,' was devised by Harten et al. (1983). The Riemann problem is approximated as consisting of only three constant states—$\mathcal{U}_{\rm L}$, $\mathcal{U}_*$, and $\mathcal{U}_{\rm R}$—separated by two waves[1] (cf. left panel in Figure 4). Application of Equation (13) across the two waves gives

$$-\alpha_-^q\left(\mathcal{U}_{\rm L} - \mathcal{U}_*\right) = \mathcal{F}^q\left(\mathcal{U}_{\rm L}\right) - \mathcal{F}^q\left(\mathcal{U}_*\right), \tag{14}$$

$$\alpha_+^q\left(\mathcal{U}_* - \mathcal{U}_{\rm R}\right) = \mathcal{F}^q\left(\mathcal{U}_*\right) - \mathcal{F}^q\left(\mathcal{U}_{\rm R}\right). \tag{15}$$

Here $\mathcal{U}_{\rm L}$ and $\mathcal{U}_{\rm R}$ are the reconstructed face values on either side of a cell interface, as in Equations (8)-(11), and $\alpha_\pm^q = \max\left[0, \pm\lambda_\pm^q\left(\mathcal{U}_{\rm L}\right), \pm\lambda_\pm^q\left(\mathcal{U}_{\rm R}\right)\right]$ are the fastest left- $(-)$ and right- $(+)$ moving hyperbolic wave speeds (magnitudes of eigenvalues of $\partial\mathcal{F}^q/\partial\mathcal{U}$). As the expression for $\alpha_\pm^q$ indicates, the wave speed estimates are computed from values on the left and right side of the interface, and the largest of the two is used in Equations (14) and (15) (cf. Davis 1988). Note that $\alpha_\pm^q \geq 0$. Solving Equations (14) and (15) for $\mathcal{U}_{\rm HLL} = \mathcal{U}_*$ and $\mathcal{F}_{\rm HLL}^q = \mathcal{F}^q\left(\mathcal{U}_*\right)$ yields

$$\mathcal{U}_{\rm HLL} = \frac{\alpha_+^q\,\mathcal{U}_{\rm R} + \alpha_-^q\,\mathcal{U}_{\rm L} - \left[\mathcal{F}^q\left(\mathcal{U}_{\rm R}\right) - \mathcal{F}^q\left(\mathcal{U}_{\rm L}\right)\right]}{\alpha_+^q + \alpha_-^q}, \tag{16}$$

$$\mathcal{F}_{\rm HLL}^q = \frac{\alpha_+^q\,\mathcal{F}^q\left(\mathcal{U}_{\rm L}\right) + \alpha_-^q\,\mathcal{F}^q\left(\mathcal{U}_{\rm R}\right) - \alpha_+^q\alpha_-^q\left(\mathcal{U}_{\rm R} - \mathcal{U}_{\rm L}\right)}{\alpha_+^q + \alpha_-^q}. \tag{17}$$

[1] Here we assume that at least two wave speeds can be associated with the hyperbolic system represented by Equation (7).

When this solver is selected, it is $\mathcal{F}^q_{\rm HLL}$ that goes into the cell-averaged balance equation in Equation (2). Note that these expressions are valid for sub- *and* supersonic flows. (Here, by supersonic we mean that the eigenvalues of $\partial\mathcal{F}^q/\partial\mathcal{U}$ are either all positive or all negative. For subsonic flows there are both positive and negative eigenvalues.) For 'supersonic' flow to the left ($\alpha^q_+ = 0$) or to the right ($\alpha^q_- = 0$) the HLL flux reduces to a pure upwind flux (i.e., information from only one side of the interface is used to compute the flux). For an isolated shock wave, the HLL approximate Riemann solver agrees with the true solution to the Riemann problem. For 'subsonic' flows, the third term on the right-hand-side of (17) acts as a diffusion term which damps out grid-scale oscillations. This term, which is proportional to the jump ($\mathcal{U}_{\rm R} - \mathcal{U}_{\rm L}$), can often lead to excessive numerical dissipation of the wave types omitted in the simplification of the hyperbolic wave structure. (For hydrodynamics, these wave types include contact and shear waves; see Section 4 for some examples). Furthermore, if we set $\alpha^q_+ = \alpha^q_- = \alpha_q = \max(\alpha^q_-, \alpha^q_+)$, Equations (16) and (17) reduce to the corresponding Rusanov or local Lax-Friedrichs (LLF) average state and flux. The LLF flux assumes symmetric left- and right-moving characteristics and results in an even more dissipative scheme than the HLL flux.

The second variant we report here is known as the HLLC solver (Toro et al. 1994). In this case a third wave, which we here call the 'middle wave,' is included in addition to the fastest left- and right-moving waves entering into the HLL solver. Thus the Riemann problem has four constant states: $\mathcal{U}_{\rm L}$, $\mathcal{U}_{*\rm L}$, $\mathcal{U}_{*\rm R}$, and $\mathcal{U}_{\rm R}$, with the middle wave separating the middle states $\mathcal{U}_{*\rm L}$ and $\mathcal{U}_{*\rm R}$ (cf. right panel in Figure 4). The jump conditions at the three waves are

$$-\alpha^q_- \left(\mathcal{U}_{\rm L} - \mathcal{U}_{*\rm L}\right) = \mathcal{F}^q\left(\mathcal{U}_{\rm L}\right) - \mathcal{F}^q\left(\mathcal{U}_{*\rm L}\right), \tag{18}$$

$$\alpha^q_m \left(\mathcal{U}_{*\rm L} - \mathcal{U}_{*\rm R}\right) = \mathcal{F}^q\left(\mathcal{U}_{*\rm L}\right) - \mathcal{F}^q\left(\mathcal{U}_{*\rm R}\right), \tag{19}$$

$$\alpha^q_+ \left(\mathcal{U}_{*\rm R} - \mathcal{U}_{\rm R}\right) = \mathcal{F}^q\left(\mathcal{U}_{*\rm R}\right) - \mathcal{F}^q\left(\mathcal{U}_{\rm R}\right). \tag{20}$$

Here $\alpha^q_\pm$ are the same as in the HLL case, and α^q_m is the middle wave speed estimate, which can be positive or negative (unlike $\alpha^q_\pm$, which are restricted to non-negative values). Unlike the system of Equations (14) and (15) in the HLL case, the present system of Equations (18)-(20) is underdetermined: there are now four (sets of) unknowns $[\mathcal{U}_{*\rm L}, \mathcal{U}_{*\rm R}, \mathcal{F}^q\left(\mathcal{U}_{*\rm L}\right), \mathcal{F}^q\left(\mathcal{U}_{*\rm R}\right)]$ but only three (sets of) equations given by the jump conditions. Additional relations must be introduced, the details of which necessarily depend upon the particular system. (A concrete example is given in the case of hydrodynamics discussed in Section 3.2, in which the starting point is to use the HLL states of Equation (16) to estimate the middle wave speed α^q_m.) Once this has been done and $\mathcal{F}^q\left(\mathcal{U}_{*\rm L}\right)$ and $\mathcal{F}^q\left(\mathcal{U}_{*\rm R}\right)$ have been obtained, the HLLC flux is given by

$$\mathcal{F}^q_{\rm HLLC} = \begin{cases} \mathcal{F}^q\left(\mathcal{U}_{\rm L}\right) & \text{if } \alpha^q_- = 0, \\ \mathcal{F}^q\left(\mathcal{U}_{*\rm L}\right) & \text{if } \alpha^q_m \geq 0, \\ \mathcal{F}^q\left(\mathcal{U}_{*\rm R}\right) & \text{if } \alpha^q_m < 0, \\ \mathcal{F}^q\left(\mathcal{U}_{\rm R}\right) & \text{if } \alpha^q_+ = 0. \end{cases} \tag{21}$$

When this solver is selected, it is $\mathcal{F}^q_{\rm HLLC}$ that goes into the cell-averaged balance equation in Equation (2). As in the HLL case, the HLLC flux reduces to an upwind scheme for supersonic flows to the left ($\alpha^q_- = 0$) or to the right ($\alpha^q_+ = 0$).

Figure 4 sketches examples of so-called 'Riemann fans' for the HLL (left panel) and HLLC (right panel) approximate Riemann solvers. These are spacetime diagrams showing the trajectories of the waves and labeling the states between them. These examples illustrate subsonic flow, with $\alpha_m > 0$ in the HLLC case.

Necessary storage and the basic outlines of the HLL and HLLC solvers are implemented in classes found in the `RiemannSolvers` division of `BalanceEquations` (see the right diagram in Figure 2). Some details, including wave speed estimates (the α's) and the explicit functional forms of the fluxes $\mathcal{F}(\mathcal{U})$ needed on the right-hand sides of Equations (17) and (21)—as well as additional expressions needed in the case of the HLLC solver—depend on the

particular system being solved. These are handled by the deferred method `ComputeRiemannSolverInput` of the `abstract` class `BalancedVariableGroupTemplate`. Details relevant to the nonrelativistic test problems in this paper are given in Section 3.2.

2.1.3. BalanceEquationUpdates

Our implementation of reconstruction, flux computation (Riemann solve), and assembly of the right-hand side of Equation (2) is written to address one level of our multilevel grid structure at a time; that is, one 'mesh' in a 'chart' at a time (see Paper I). There are however two ways in which better information from a next finer level is utilized. One of these is applied in the present layer of implementation: for cells on the coarse side of a coarse/fine boundary with the next level, flux updates from faces on the coarse/fine boundary computed by and restricted (i.e., averaged) from the finer level replace those computed at the present (coarser) level. Aside from making use of more highly resolved data, this ensures conservative evolution on the union of leaf cells of all levels (modulo source terms). The second use of information from the next finer level is applied at a higher layer of coding discussed in more detail later: non-leaf cells have their updated balanced variables replaced by restriction from the next finer level (see Section 3.3). This restriction does not have direct consequences for global conservation checks, because these are tallied only from leaf cells; nevertheless this step is performed for the sake of consistency between levels.

In pursuit of efficient performance we seek to both (a) overlap work and communication in a message-passing parallel environment, and (b) work on data stored in contiguous memory when possible. To accomplish these goals we identify two types of cells. 'Exchange' cells are cells near process boundaries whose data must be communicated (or exchanged) to fill the ghost cells on neighboring processes in a 'ghost exchange' during the update of the hyperbolic balance equations. (The ghost cells of a given process are exchange cells on neighboring processes; see Paper I.) 'Non-exchange' cells, in contrast, do not have to communicate their data to neighboring processes. Goals (a) and (b) above can be accomplished by processing exchange cells and non-exchange cells separately in some instances, and by packing data from these two sets of cells into contiguous memory before certain operations. Storage we refer to as 'unpacked' is the normal mesh-oriented storage discussed in Paper I. It is this type of storage in which connectivity information culled from the oct-tree is available, i.e. location in the mesh and knowledge of siblings. Operations that need this information must be performed with unpacked storage. These include exchange of ghost cell data, and operations across cell faces that require data from sibling cells, in particular calculations of differences and fluxes. 'Packed' storage, on the other hand, has no connectivity information. It provides for efficient intra-cell operations by furnishing sequential memory access to sets of data—in particular, data only from exchange cells or data only from non-exchange cells—that are discontiguous in unpacked storage. (Data packing and unpacking is performed with the class `PackedVariableGroupForm` mentioned in Paper I.)

Algorithm 1 outlines a balance equation update on a single level. It is implemented by classes in the `BalanceEquationUpdates` division of `BalanceEquations` (see the right diagram of Figure 2). Uses of packed and unpacked storage are indicated by (P) and (U) respectively. Two major loops are explicitly shown; each of these performs computations on exchange cells, begins non-blocking communication, and then performs computations on non-exchange cells while communication continues in the background. In Algorithm 1 and later listings, right-arrows ($\rightarrow$) indicate the flow of various operations (e.g. *Compute*, *Pack*, and *Unpack*), with the input on the left and the result on the right.[2] Application of the updates to obtain new values of the variables is deferred to the time

[2]For example, packed finite differences are computed from unpacked primitive variables in line 2; primitive variables in unpacked storage are copied into packed storage in line 3; packed reconstructed primitive variables are computed from packed primitive variables and finite differences in line 4; and so on.

stepper (see Section 2.2). The slowest parts of the algorithm are those that involve both packed and unpacked storage, as these necessarily involve indirect indexing of the unpacked storage. These include computations across cell faces (differences and fluxes) and moving data between packed and unpacked storage. Ideally, the efficiency of operations in packed storage makes up for the overhead of data movement between the two storage types.

Algorithm 1 Single-level balance equation update

for iCells = EXCHANGE, NON-EXCHANGE **do**
  Compute: Primitive (U) → Differences (P)
  Pack: Primitive (U) → Primitive (P)
  Compute: Primitive (P), Differences (P) → Reconstructed Primitive (P)
  Compute: Reconstructed Primitive (P) → Reconstructed Balanced (P), Reconstructed Auxiliary (P)
  Unpack: Reconstructed (P) → Reconstructed (U)
  if iCells == EXCHANGE **then**
    Begin Exchange: Reconstructed (U)
  end if
end for
Apply Boundary Conditions: Reconstructed Interior (U) → Reconstructed Exterior (U)
Finish Exchange: Reconstructed (U)
Compute: Reconstructed (U) → Riemann Solver Input (U)
for iCells = EXCHANGE, NON-EXCHANGE **do**
  Compute: Riemann Solver Input (U) → Fluxes (P)
  Compute: Fluxes (P), Sources (P) → Updates (P)
  Unpack: Updates (P) → Updates (U)
  if iCells == EXCHANGE **then**
    Begin Exchange: Updates (U)
  end if
end for
Finish Exchange: Updates (U)

2.2. Steps

As discussed in connection with Equation (2), a spatially discretized system with a continuous time derivative can be considered a system of time-dependent ordinary differential equations. We write this as

$$\frac{d\langle \mathcal{U} \rangle}{dt} = \mathcal{L}\left[\langle \mathcal{U} \rangle\right], \tag{22}$$

where (for example) in the case of balance equations $\mathcal{L}\left[\langle \mathcal{U} \rangle\right]$ is given by the right-hand side of Equation (2). The system consists of one equation—or rather, one set of equations, since $\mathcal{U}$ is a vector—for $\langle \mathcal{U} \rangle$ in every cell. (We suppress indexing not only over the components of $\mathcal{U}$, but also over cells, taking this to be implied by the volume-average-denoting angle brackets.) Equation (22) is amenable to discretization and integration in time using standard explicit techniques. (By 'explicit' we mean that the solution at time t^{n+1} depends only on values known at t^n, where the sans-parentheses superscript n denotes a value at the beginning of the nth time step.)

A basic building block of many such integration algorithms is a single update

$$\mathcal{K}^{(i)} = \Delta t\, \mathcal{L}\left[\langle \mathcal{U} \rangle^{(i-1)}\right], \tag{23}$$

where Δt is the full time step. Stable and accurate schemes typically involve multiple such updates or 'substeps,' indexed by the parenthetical superscript (i), with higher-order schemes requiring more updates. Typically $\langle \mathcal{U} \rangle^{(0)} = \langle \mathcal{U} \rangle^n$ on the right-hand side of Equation (23) for the first update $\mathcal{K}^{(1)}$. Subsequent intermediate values $\langle \mathcal{U} \rangle^{(i)}$ then depend on prior updates.

In all the test problems in this paper we use a second-order Total Variation Diminishing (TVD) Runge-Kutta step (e.g. Shu 1998, also known as Heun's method) consisting of two substeps. The two updates $\mathcal{K}^{(1)}$ and $\mathcal{K}^{(2)}$ are obtained by respectively using

$$\langle \mathcal{U} \rangle^{(0)} = \langle \mathcal{U} \rangle^n, \tag{24}$$

$$\langle \mathcal{U} \rangle^{(1)} = \langle \mathcal{U} \rangle^n + \mathcal{K}^{(1)} \tag{25}$$

on the right-hand side of Equation (23). Then

$$\langle \mathcal{U} \rangle^{n+1} = \langle \mathcal{U} \rangle^n + \frac{1}{2} \left(\mathcal{K}^{(1)} + \mathcal{K}^{(2)} \right) \tag{26}$$

is the solution at t^{n+1}.

There are a couple of points to make in connection with balance equations discussed in Section 2.1. First, in this context $\langle \mathcal{U} \rangle^{n+1}$ are the balanced variables, from which the primitive and auxiliary variables $\langle \mathcal{W} \rangle^{n+1}$ and $\langle \mathcal{A} \rangle^{n+1}$ are subsequently obtained. These operations, and also the computations in Equations (24)-(26), are all performed on all proper and ghost cells of the Interior submesh: the algorithm implementing computation of an update $\mathcal{K}$ based on Equation (2), given at the end of Section 2.1, already includes a ghost exchange of $\mathcal{K}$. (See Paper I for more on proper and ghost cells and the Interior submesh.) Thus no additional communication is needed at this point to complete the step. Moreover, being performed over all Interior cells, application of the updates works on contiguous memory even though it is 'unpacked' storage. Second, flux updates at faces on the coarse/fine boundary with a coarser level are recorded in a manner consistent with the update scheme described above, in order that they can be utilized when the next coarser level is evolved (as discussed at the beginning of Section 2.1.3).

Classes implementing this approach are found in the `Steps` division of `Solvers` (see the middle diagram in Figure 2). They provide a method (that is, a subroutine) that accepts a time step $\Delta t = t^{n+1} - t^n$ as an argument and performs a single step from t^n to t^{n+1}—consisting, as we have seen, of substeps—on a single level of our multilevel grid (i.e., on a single mesh in a chart; see Paper I). Time step determination, addressing of multiple levels, and looping over many steps are implemented in a higher layer of coding and are further discussed later (Section 3.3).

3. PHYSICS

After `Basics` (discussed in Paper I) and `Mathematics` (whose divisions `Manifolds` and `Solvers` have been discussed in Paper I and Section 2 of the present paper respectively), the third major division of GenASiS is `Physics` (see Figure 1). As illustrated in Figure 5, `Physics` is subdivided into `Spaces`, `StressEnergies`, and `Universes`. In this paper we discuss the `Galilean` division of `Spaces`, the `Fluids` division of `StressEnergies`, and the class `FreeFluidForm` in `Universes`.

3.1. Galilean

The `Spaces` division of `Physics` contains classes implementing physical spaces associated with different descriptions of spacetime, including treatments of gravity. The problems presented in this paper utilize only a `Galilean`

space that is nonrelativistic and has no gravity. The `Galilean` division of `Spaces` (see Figure 5) contains extensions of `GeometryBasicForm`, `ChartForm`, and `AtlasForm` from the `Manifolds` division of `Mathematics` (see Paper I). In this part of the code we replace the mathematical terms Chart and Atlas with the more physics-friendly 'CoordinatePatch' and 'Space' respectively. The Galilean 'no-gravity' classes are trivial extensions serving only to change to more physics-friendly nomenclature, but classes we will report on in future papers have additional functionality to accommodate gravity.

3.2. Fluids

`Fluids` is the only division of `StressEnergies` we report on in this paper (see Figure 5). We describe only ideal fluids; dissipative processes (e.g. viscosity and heat conduction) are not included, and the equations describe adiabatic flows: $\partial s/\partial t + \mathbf{v}\cdot\nabla s \equiv ds/dt = 0$, where s is the entropy per baryon (see for example Landau and Lifshitz 1959, for an introduction to fluid mechanics). The major classes we discuss here are `DustForm`, `FluidTemplate`, and `FluidPolytropicForm`.

'Dust' may be regarded as a pressureless fluid, but in fact it is 'less than a fluid' in the sense that no pressure or energy variables are needed to describe it. The class `DustForm` is an extension of `BalancedVariableGroupTemplate` (see Section 2.1). Its balanced variables $\mathcal{U}$ are the conserved baryon density D and the momentum density $\mathbf{S}$. The primitive variables $\mathcal{W}$ are the comoving baryon density n and the three-velocity $\mathbf{v}$. The only auxiliary variable $\mathcal{A}$ is the average baryon mass m. The relations between these variables are

$$D = n, \tag{27}$$
$$\mathbf{S} = mn\mathbf{v}. \tag{28}$$

The baryon and momentum fluxes $\mathcal{F}(\mathcal{U})$ are

$$\mathcal{F}(D) = n\mathbf{v}, \tag{29}$$
$$\mathcal{F}(\mathbf{S}) = mn\mathbf{v}\mathbf{v}. \tag{30}$$

The wave velocities (eigenvalues of $\partial\mathcal{F}^q/\partial\mathcal{U}$; see Section 2.1) are simply components of the three-velocity $\mathbf{v}$: $\lambda^q_- = \lambda^q_+ = v^q$ for Equation (7) in the q dimension. Recall that $\lambda^q_\pm$ are used to obtain estimates of the left- and right-moving wave speeds needed by the HLL solver (Section 2.1). Note that for Dust the flow is perforce 'supersonic' (i.e. either α^q_+ or α^q_- vanishes, depending on the sign of v^q). The HLLC solver is not relevant for Dust, which does not support additional waves.

`FluidTemplate` is an `abstract` extension of `DustForm` that provides the basic infrastructure for a fluid with pressure and internal energy without committing to a particular equation of state. In addition to the variables defined for Dust, there is the balanced energy density G (an additional balanced variable in $\mathcal{U}$), the internal energy density e (an additional primitive variable in $\mathcal{W}$), and the pressure p (an additional auxiliary variable in $\mathcal{A}$). The balanced energy density is given by

$$G = e + \frac{1}{2}mn|\mathbf{v}|^2. \tag{31}$$

We note also the momentum and energy fluxes

$$\mathcal{F}(\mathbf{S}) = mn\mathbf{v}\mathbf{v} + p\mathbf{I}, \tag{32}$$
$$\mathcal{F}(G) = \left(e + p + \frac{1}{2}mn|\mathbf{v}|^2\right)\mathbf{v}. \tag{33}$$

The momentum flux now contains an isotropic contribution from the pressure (**I** is the rank-two unit tensor); the baryon flux remains as in Equation (29). There are now three distinct wave velocities: $\lambda^q_- = v^q - c_s$, $\lambda^q_m = v^q$, and $\lambda^q_+ = v^q + c_s$, where the adiabatic sound speed $c_s = \sqrt{(\partial p/\partial \rho)_s} = \sqrt{(\Gamma_s p/\rho)}$, with adiabatic index $\Gamma_s \equiv (\partial \ln p/\partial \ln \rho)_s$ and mass density $\rho = mn$. The eigenvalues λ^q_- and λ^q_+ are propagation velocities associated with acoustic waves, while λ^q_m is the propagation velocity of contact and shear waves. (The contact wave is also referred to as the entropy wave.) The HLL solver uses only the largest-magnitude wave velocities $\lambda^q_\pm$. Moreover, recall (Section 2.1) that in the supersonic case (either $\alpha^q_+ = 0$ or $\alpha^q_- = 0$) the HLLC solver yields the same result as the HLL solver, namely, upwind fluxes computed solely from the reconstructed values on either the left or right side of the interface (see Equation (21)). For subsonic flows the HLLC solver uses, in addition, an estimate of the middle wave velocity $\alpha^q_m = \lambda^q_m$. We follow Batten et al. (1997) and form the middle wave speed from the mass density and momentum density of the HLL average state (cf. our Equations (16), (27), (28)):

$$\alpha^q_m = \frac{S^q_{\rm HLL}}{mD_{\rm HLL}} = \frac{S^q_{\rm R}\left(\alpha^q_+ - v^q_{\rm R}\right) + S^q_{\rm L}\left(\alpha^q_- + v^q_{\rm L}\right) - (p_{\rm R} - p_{\rm L})}{mD_{\rm R}\left(\alpha^q_+ - v^q_{\rm R}\right) + mD_{\rm L}\left(\alpha^q_- + v^q_{\rm L}\right)}. \tag{34}$$

Recall that the three waves in the HLLC framework separate four constant states: the reconstructed values $\mathcal{U}_{\rm L}$ and $\mathcal{U}_{\rm R}$, and the middle states $\mathcal{U}_{*\rm L}$ and $\mathcal{U}_{*\rm R}$ separated by the middle wave (see Fig. 4). In the subsonic case the fluxes are constructed from either $\mathcal{U}_{*\rm L}$ or $\mathcal{U}_{*\rm R}$, depending on the sign of α^q_m (Equation (21)). Obtaining $\mathcal{U}_{*\rm L}$ or $\mathcal{U}_{*\rm R}$ in terms of the known reconstructed values $\mathcal{U}_{\rm L}$ or $\mathcal{U}_{\rm R}$ is accomplished by making a key assumption followed by use of the jump conditions of Equations (18) or (20) across the outer waves. The key assumption is to set the normal velocity component equal to the estimated middle wave speed,

$$v^q_{*\rm L/R} = \alpha^q_m, \tag{35}$$

for both the left and right middle states. The density jump conditions give

$$D_{*\rm L/R} = \frac{\alpha^q_\mp \pm v^q_{\rm L/R}}{\alpha^q_\mp \pm \alpha^q_m} D_{\rm L/R}, \tag{36}$$

with the upper and lower signs corresponding to the left (L) and right (R) versions of the equation respectively. Using this in conjunction with the transverse components of the momentum jump conditions yields

$$v^{r\neq q}_{*\rm L/R} = v^{r\neq q}_{\rm L/R}, \tag{37}$$

while the normal component produces

$$p_{*\rm L/R} = p_{\rm L/R} - D_{\rm L/R}\left(\alpha^q_\mp \pm v^q_{\rm L/R}\right)\left(\alpha^q_m \pm v^q_{\rm L/R}\right). \tag{38}$$

The energy jump conditions give

$$G_{*\rm L/R} = \frac{\left(\alpha^q_\mp \pm v^q_{\rm L/R}\right) G_{*\rm L/R} \pm v^q_{\rm L/R} p_{\rm L/R} \mp \alpha^q_m p_{*\rm L/R}}{\alpha^q_\mp \pm \alpha^q_m}. \tag{39}$$

Equations (34)-(39) provide the necessary and sufficient information needed to build the HLLC flux. As an aside, it is not immediately apparent from Equation (38), but it turns out that enforcing the normal velocity to be constant across the middle wave in the Riemann fan (cf. Equation (35)) implies that the pressure is also constant across the middle wave. This is because the normal component of the momentum jump condition across the middle wave (Equation (19)) yields

$$p_{*\rm L} + mD_{*\rm L} v^q_{*\rm L}\left(v^q_{*\rm L} - \alpha^q_m\right) = p_{*\rm R} + mD_{*\rm R} v^q_{*\rm R}\left(v^q_{*\rm R} - \alpha^q_m\right). \tag{40}$$

It then follows from Equation (35) that $p_{*\mathrm{L}} = p_{*\mathrm{R}}$.

`FluidPolytropicForm` is an extension of `FluidTemplate` that furnishes an equation of state of the form

$$p = \kappa n^{\gamma}. \tag{41}$$

From the first law of thermodynamics for an ideal fluid,

$$de = \frac{e+p}{n}\, dn, \tag{42}$$

it follows that $p = (\gamma - 1)\, e$ and that the adiabatic index $\Gamma_s = \gamma$. The adiabatic index γ is a constant parameter. In the absence of energy source terms, the polytropic 'constant' κ remains constant unless shocks generate dissipation (captured automatically by the finite-volume approach with HLL or HLLC solvers; cf. Section 2.1).

The class `DustForm` and its extensions `FluidTemplate` and `FluidPolytropicForm` contain storage corresponding to a single mesh, which is a single level of a refinable multilevel coordinate patch (or chart in mathematical parlance; see Section 3.1 and Paper I). The class `FluidCoordinatePatchForm` contains an array of `class ( DustForm )` corresponding to the refinement levels of a CoordinatePatch. This array, being polymorphic, can be allocated to any type of fluid that is an extension of `DustForm`. Similarly, `FluidSpaceForm` contains an array of `FluidCoordinatePatchForm` corresponding to the CoordinatePatches comprising a Space (or Atlas in mathematical jargon; again, see Section 3.1 and Paper I).

3.3. FreeFluidForm

The classes in the `Universes` division of `Physics` (see Figure 5) include a physical space and one or more forms of stress-energy. Reporting in this paper only a `Galilean` space without gravity (Section 3.1), and `Fluids` as the only form of stress energy (Section 3.2), the class `FreeFluidForm` is the only type of 'universe' we describe here.

The central functionality of any Universe, including an instance of `FreeFluidForm`, is its method `Evolve`. In basic outline the evolution is simply a loop over individual time steps covering the interval from parameters `StartTime` to `EndTime`; see Algorithm 2 for `FreeFluidForm % Evolve`. However, there are complications from multiple levels of refinement.[3] Our multilevel explicit evolution features 'subcycling' of deeper levels, or 'refinement in time' as well as space. The first hint of this appears in line 3: there is a global `Time` for the coordinate patch as a whole, but also an array `LevelTime` that tracks the time to which each level has been evolved. At the beginning of a global time step all elements of `LevelTime` are initialized to the global `Time`. Subcycling also figures into the computation of `TimeStep` for the coarsest level (line 4): because each successive level features cell subdivision by a factor of two in each dimension, the time step differs by a factor of two between adjacent levels, and so the time step for the deepest level is scaled up by a factor of $2^{\texttt{nLevels}-1}$ to yield the time step for the coarsest level (`iLevel` $= 1$). For balance equations with non-stiff sources (i.e., the fastest characteristic time scale is set by the flux divergence term in Equation (1)), the time step Δt is determined from the Courant-Friedrichs-Lewy (CFL) condition for stable explicit time stepping; $\Delta t = \mathrm{C}_{\mathrm{CFL}} \times \Delta t_{\mathrm{CFL}}$, where $\mathrm{C}_{\mathrm{CFL}} \lesssim 1$ is the 'Courant factor' and

$$\Delta t_{\mathrm{CFL}} = \min_{\mathrm{cells}} \left[\min_{q} \left(\frac{\Delta q}{\max(|\lambda_q^-|, |\lambda_q^+|)} \right) \right] \tag{43}$$

[3] There are also complications—which we do not discuss here—in the case of a Space with multiple CoordinatePatches (or, in mathematical terms, an Atlas with multiple Charts).

is the Courant time step. A call to `EvolveLevel` for the coarsest level (line 5) recursively evolves all the deeper levels, updating the elements of `LevelTime` in the process. All levels are synchronized by the time this call returns, so that the global `Time` is updated to the element of `LevelTime` corresponding to the coarsest level (line 6).

The method `FreeFluidForm % EvolveLevel` is very simple (see Algorithm 3): the fact that two steps of Level i are performed for every step of Level $i-1$ makes it convenient to have this method simply group two successive calls to a more primitive method `StepLevel` (lines 4 and 8). Two `if` blocks round out this method's functionality. The one at the top (lines 1-3) terminates the recursion to deeper levels by returning when the deepest level has been reached. The one between calls to `StepLevel` (lines 5-7) returns if `iLevel == 1`; i.e., only a single step of the coarsest level is performed. The only other wrinkle is the optional argument `ForParentsThisOption`, which was not included in the top-level call from `FreeFluidForm % Evolve`. This allows updates at coarse/fine boundaries to be accumulated between the two steps at Level i for use at Level $i-1$.

The time stepper described in Section 2.2 is finally invoked for Level i in the method `FreeFluidForm % StepLevel` (Algorithm 4), and this is the method's main purpose; but most of this routine (in terms of lines of code) is actually dedicated to interactions between adjacent levels. Lines 1-6 of Algorithm 4 show that if a deeper level exists, that level is evolved *before* the current level. Boundary values (i.e. data for the Exterior submesh; see Paper I) of Level $i+1$ are filled in by prolongation from Level i (line 2). Prolongation is the interpolation operation that provides data for finer cells from coarser cells (e.g. after refinement; see Paper I for details on how prolongation is done in GenASiS). In line 3, the variable `ForParentsNext` is allocated to hold updates at coarse/fine boundaries to be computed on Level $i+1$. The call to `EvolveLevel` $i+1$ follows on line 4. Its time step argument is half that to be used on Level i; but as discussed in the previous paragraph, two steps at Level $i+1$ will be taken by `EvolveLevel`. The call also includes the optional argument containing temporary storage `ForParentsNext` in which the updates at the coarse/fine boundary between Levels i and $i+1$ are accumulated. After the call returns, these updates are restricted from Level $i+1$ to the member `FromChildren` of the Level i stepper (line 5). Restriction is the averaging procedure that provides data for a coarse cell from the finer cells it encompasses (see Paper I for details on how restriction is done in GenASiS). When the stepper at Level i takes a time step (line 7), these updates `FromChildren` overwrite those computed from Level i at the coarse/fine boundary with those from Level $i+1$, ensuring conservative evolution at this boundary with the more accurate update computed at the finer level. After the Level i step, consistency with Level $i+1$ is completed by restriction of the fluid from Level $i+1$ to Level i in cells where these levels overlap (lines 8-10). The Level i step is now complete, and the corresponding element of `LevelTime` is updated to reflect the advance by `TimeStep`. The final task is to accumulate, in the optional argument `ForParentsThisOption`, the coarse/fine boundary updates stored in the Level i stepper member `ForParents`, which will be used later by the Level $i-1$ stepper to handle updates at the coarse/fine boundary between Levels $i-1$ and i.

We now briefly summarize the recursive multilevel evolution executed by the methods `Evolve`, `EvolveLevel`, and `StepLevel` of `FreeFluidForm`. Each iteration of the loop in `Evolve` takes a single global time step by calling the method `EvolveLevel` for the coarsest level. However, the coarsest level is not immediately stepped forward. Instead, `EvolveLevel` and `StepLevel` recursively work their way down to the deepest, most refined level, which is the first to be evolved. From the bottom up, two steps of each Level $i+1$ are performed for each step at Level i. By construction, the evolution of the entire multilevel structure ends up fully synchronized once this process works its way up to complete a single step at the coarsest level. The conservation (in the absence of sources, as in most of the test problems reported in this paper) implied by the divergence structure of the balance equations is ensured by using, at Level i, the updates at coarse/fine boundaries computed at Level $i+1$. Further—less critical and perhaps fastidious—consistency between levels is achieved by restriction from Level $i+1$ to Level i where these overlap. Our algorithm for time integration of the balance equations on a multilevel grid is demonstrated in the 2D and 3D Sedov-Taylor blast wave problems (Section 4.2.3).

Algorithm 2 FreeFluidForm % Evolve

Set: Time = StartTime
while Time $<$ EndTime **do**
  Set: LevelTime (1 : nLevels) = Time
  Compute: TimeStep for iLevel = 1
  Call: EvolveLevel (LevelTime, TimeStep, iLevel = 1)
  Set: Time = LevelTime (1)
end while

Algorithm 3 FreeFluidForm % EvolveLevel (LevelTime, TimeStep, iLevel,ForParentsThisOption)

if iLevel $>$ nLevels **then**
  Return
end if
Call: StepLevel (LevelTime, TimeStep, iLevel, ForParentsThisOption)
if iLevel $==$ 1 **then**
  Return
end if
Call: StepLevel (LevelTime, TimeStep, iLevel, ForParentsThisOption)

Algorithm 4 FreeFluidForm % StepLevel (LevelTime, TimeStep, iLevel, ForParentsThisOption)

if iLevel + 1$<=$ nLevels **then**
  Prolong: Fluid (iLevel), Interior $\rightarrow$ Fluid (iLevel + 1), Exterior
  Allocate: ForParentsNext
  Call: EvolveLevel (LevelTime, TimeStep / 2.0, iLevel + 1, ForParentsNext)
  Restrict: ForParentsNext, Exterior $\rightarrow$ Stepper (iLevel) % FromChildren, Interior
end if
Call: Stepper (iLevel) % Step (TimeStep)
if iLevel + 1$<=$ nLevels **then**
  Restrict: Fluid (iLevel + 1), Interior $\rightarrow$ Fluid (iLevel), Interior
end if
Compute: LevelTime (iLevel) = LevelTime (iLevel) + TimeStep
if present (ForParentsThisOption) **then**
  Compute: ForParentsThisOption = ForParentsThisOption + Stepper (iLevel) % ForParents
end if

4. APPLICATIONS

In this section we present results from numerical test problems demonstrating the capabilities of the numerical methods and algorithms implemented in GenASiS to solve the equations of nonrelativistic ideal hydrodynamics. The test problems have been chosen to validate the correctness of our implementation and to reveal the scheme's strengths and weaknesses. Most are well-known in the literature (e.g. Toro 2009; Fryxell et al. 2000; Liska and Wendroff 2003; Stone et al. 2008). The programs that initialize and evolve the test problems are found in the `Applications` division of GenASiS (see Figure 1). Each of these 'driver' programs declares an instance of `FreeFluidForm` (Section 3.3), which includes both the space and the fluid; sets up the problem's initial conditions; and calls the `FreeFluidForm` object's `Evolve` method.

By definition a test problem has an accepted solution against which program output can be compared. For some problems an analytic solution exists; in other cases no analytic solution exists, but 'known' numerical solutions are available in the literature. In the cases where no analytic solution is available, it is also common practice to compare low-resolution results with a high-resolution reference solution (self-convergence; e.g. Stone et al. 1992), which we do in Section 4.2.2. For test problems where an analytic solution χ is available, we quantify the quality of the numerical solution $\langle\chi\rangle$ produced with GenASiS by computing the relative L_1 error norm

$$L_1\left(\chi, N\right) = \frac{\sum_{\text{cells}} |\langle\chi\rangle - \chi|}{\sum_{\text{cells}} |\chi|}, \tag{44}$$

where the sums extend over all N cells covering the computational domain. For a numerical method of spatial order p, the error $|\langle\chi\rangle - \chi|$ decreases with the mesh spacing as $(\Delta q)^p \propto N_q^{-p}$, where N_q is the number of cells in the qth coordinate direction (assuming a uniform mesh). A similar argument holds for the temporal order of the numerical method (due to the CFL condition (Equation 43), the ratio $\Delta t/\Delta q$ remains fixed as the mesh spacing decreases), and the temporal error decreases with decreasing mesh spacing at a rate determined by the temporal order of the scheme. We determine the formal order of the numerical method by computing the relative L_1 error norm in a resolution study: by evolving an initial condition to some specified end time with multiple grid resolutions (e.g. $N_{q,1} < N_{q,2}$), the formal order of the scheme p, or the rate at which the numerical solution approaches the analytic solution as the number of grid cells increases from $N_{q,1}$ to $N_{q,2}$, is determined from

$$-p = \frac{\log\left[L_1(\chi, N_{q,1})/L_1(\chi, N_{q,2})\right]}{\log\left[N_{q,1}/N_{q,2}\right]}. \tag{45}$$

The hydrodynamics scheme implemented in GenASiS is designed to be second-order accurate (we use linear interpolation in space and evolve the resulting system of ODEs with a second-order Runge-Kutta time integrator), and we expect second-order convergence for smooth flows. For flows containing discontinuities, the scheme switches to constant spatial interpolation in the vicinity of the discontinuities (cf. Equation (6)), and the formal order of the scheme reduces to first-order.

Figure 6 shows that the `Applications` division of GenASiS includes `SmoothFluidTests`, `DiscontinuousFluidTests`, and `FluidInstabilityTests`. Our test results illustrate the basic competence of our implementation, demonstrate the manifest superiority of the HLLC over the HLL Riemann solver in a number of interesting cases, and provide preliminary indications of the code's ability to scale and to function with cell-by-cell fixed-mesh refinement. We present results problems in one (1D), two (2D), and three (3D) space dimensions. For the 1D and 2D test problems we set the Courant factor to $\mathrm{C_{CFL}} = 0.5$, and for the 3D tests we use $\mathrm{C_{CFL}} = 0.3$.

4.1. SmoothFluidTests

The tests in the `SmoothFluidTests` division of `Applications` (see the right diagram in Figure 6), which have smooth analytic solutions, enable us to determine the order of accuracy of the hydrodynamics solvers implemented in GenASiS. The driver programs in `SmoothFluidTests` are `FluidAdvection` and `LinearFluidWaves`. We compare the numerical solutions with the analytic solution for multiple grid resolutions, compute the relative L_1 error norm (Equation (44)), and calculate the rate p (Equation (45)) at which the error norm decreases with increasing grid resolution. We demonstrate second-order accuracy with both the HLL and the HLLC solvers.

4.1.1. FluidAdvection

In this test a smooth density profile is advected with a constant velocity field. A constant pressure field is also present, and we evolve the full system of hydrodynamics equations, not just the continuity equation; thus the tests in this section involve the entropy (or contact) wave. We present and compare results obtained with the HLL and HLLC Riemann solvers, and investigate the sensitivity of the results to the flow Mach number $\mathrm{Ma} = |\mathbf{v}|/c_s$. It is reasonable to expect that the results obtained with the HLL solver are sensitive to Ma, since this Riemann solver considers only the acoustic waves in the Riemann fan, and ignores the entropy wave. This is especially true for highly subsonic flows where the entropy wave is clearly separated from the acoustic waves in the Riemann fan (cf. Figure 4). Periodic boundary conditions are used in all the tests presented in this section. We show 1D and 2D results.

In the 1D tests the computational domain is confined to $x \in [0, 1]$. The initial density is set to $\rho = \rho_0 + \delta\rho \times \sin(2\pi x/L_x)$, with $\rho_0 = 1.0$ and $\delta\rho = 0.1$, and the only nonzero velocity component is $v_x = 1$. The (constant) pressure is parametrized by the Mach number $p = \rho_0|\mathbf{v}|^2/(\gamma \mathrm{Ma}^2)$. The adiabatic index is set to $\gamma = 1.4$. The density profile is advected across the domain five times, until $t = 5.0$.

Results from the 1D advection tests with $\mathrm{Ma} = 0.6$ are tabulated in Table 1, where we list the L_1 error norms of the density $L_1(\rho)$, for multiple grid resolutions, at $t = 1.0$ and $t = 5.0$, computed with the HLL and the HLLC Riemann solvers. We also list the rate at which the numerical solution converges to the true solution (the initial condition in this case) as the grid resolution is increased. From Table 1 we see that the numerical scheme is second-order accurate for this test, both with the HLL and the HLLC Riemann solvers. The L_1 error norms are somewhat smaller (about 20% for $N_x = 16$) when the HLLC solver is used, but the rate of convergence is slightly higher with the HLL solver, and the difference is reduced to a few percent for the highest resolution runs.

Results from the 1D advection test with $\mathrm{Ma} = 0.1$ are tabulated in Table 2. The difference between the HLL and HLLC results becomes more pronounced when the Mach number is reduced. Both Riemann solvers result in second order convergence of the L_1 error norm when $\mathrm{Ma} = 0.1$, but the errors obtained with the HLL solver are significantly larger—about a factor of two compared to the corresponding results listed in Table 1. Errors with the HLLC results are somewhat smaller than those in Table 1. In fact, the HLLC solver becomes exact for this test when the advection velocity is zero, as demonstrated with the isolated contact discontinuity Riemann problem in Section 4.2.1.

In the 2D advection tests the sine wave propagates parallel to $\mathbf{k} = (2\pi/L_x)\hat{\mathbf{x}} + (2\pi/L_y)\hat{\mathbf{y}}$, that is, with an angle $\alpha = \tan^{-1}(L_x/L_y)$ with respect to the x-axis. The computational domain is now restricted to $[x, y] \in [0, L_x]\times[1, L_y]$, the initial density is set to $\rho = \rho_0 + \delta\rho \times \sin(2\pi[x/L_x + y/L_y])$, and the velocity vector is $\mathbf{v} = [v_x, v_y, v_z] = [\cos\alpha, \sin\alpha, 0] = [L_x^{-1}, L_y^{-1}, 0]/\left(L_x^{-2} + L_y^{-2}\right)^{1/2}$. With $L_x = 2L_y$ and $L_y = \sqrt{5}/2$ the propagation angle is $\alpha \approx 63.4°$, and the sine wave returns to its initial position for $t = 1.0$. We set $\mathrm{Ma} = 0.6$ and evolve until $t = 5.0$. All other parameters are the same as in the 1D tests.

Results from the 2D advection tests at $t = 1$ and $t = 5$, computed with the HLL and HLLC Riemann solvers for different grid resolutions, are tabulated in Table 3. The results reveal a trend similar to that seen in the 1D tests: with both Riemann solvers, the numerical solution eventually converges to the analytic solution at the expected second-order rate. The errors decrease at a somewhat faster rate when the HLL solver is used, but the L_1 error norms obtained with the HLLC solver are smaller than those obtained with the HLL solver (about 7% smaller at $t = 5$ with $N_x = 512$).

4.1.2. LinearFluidWaves

Our linear wave tests are similar to those in Stone et al. (2008). We initialize a 1D periodic domain $x \in [0, 1]$ with a background state $\rho_0 = 1$, $p_0 = 3/5$. The adiabatic index is set to $\gamma = 5/3$, so that the background sound speed is $c_{s,0} = 1$. The background medium is at rest for the sound wave test, while for the shear wave tests we set $\mathbf{v} \cdot \hat{\mathbf{x}} = 0.5$ (i.e., Ma $= 0.5$). We initialize sound waves by setting $[\rho, v_x, v_y, v_z, p] = [\rho_0, 0, 0, 0, p_0] + A\,\mathbf{R}^T_{\pm c_s} \sin(2\pi x/L_x)$, while shear waves are initialized by setting $[\rho, v_x, v_y, v_z, p] = [\rho_0, 0.5, 0, 0, p_0] + A\,(\mathbf{R}^T_{v_y} + \mathbf{R}^T_{v_z}) \sin(2\pi x/L_x)$. The amplitude of the linear waves is set to $A = 10^{-6}$. When initializing the different wave types, $\mathbf{R}^T_{\pm c_s} = [c_{s,0}^{-2}, \pm c_{s,0}^{-1}\rho_0^{-1}, 0, 0, 1]$, $\mathbf{R}^T_{v_y} = [0, 0, 1, 0, 0]$, and $\mathbf{R}^T_{v_z} = [0, 0, 0, 1, 0]$ are right eigenvectors obtained from the quasi-linear form of the Euler equations in primitive variables, and are associated with left ($-$) and right ($+$) propagating sound waves, and shear waves associated with the y- and z-components of the velocity, respectively. We let the waves propagate a distance L_x (until $t = 1.0$ for sound waves and $t = 2.0$ for shear waves) and compute the L_1 error norm.

Results from the convergence tests are displayed in Figure 7, where we plot the L_1-error norm versus N_x for acoustic (left panel) and shear (right panel) waves. Results obtained with the HLL and HLLC Riemann solvers are shown in grey and black, respectively. We obtain second-order convergence in these tests. For sound waves, the results obtained with the HLL and HLLC solvers are identical, while the errors obtained with the HLLC solver are somewhat smaller for the shear wave tests (about 30% for $N_x = 16$ and about 8% for $N_x = 1024$).

4.2. DiscontinuousFluidTests

The `DiscontinuousFluidTests` division of `Applications` (see the middle right diagram in Figure 6) tests the ability of GenASiS to handle shocks and other discontinuities, which are ubiquitous in astrophysical flows. Finite-volume methods based on the integral formulation in Equation (2) are particularly well suited for flows that may develop discontinuities (e.g. LeVeque 2002). In addition to the driver program `RiemannProblem`, which is used for several test problems in one and multiple dimensions, `DiscontinuousFluidTests` also includes the `InteractingBlastWave` and `SedovTaylorBlastWave` driver programs.

4.2.1. RiemannProblem

A Riemann problem involves a hyperbolic system of equations (e.g. the Euler equations for gas dynamics) and a set of piecewise constant initial data separated into left (L) and right (R) states by an initial discontinuity. The solution for $t > 0$ typically consists of a finite set of waves propagating away from the initial location of the discontinuity. The adiabatic index is set to $\gamma = 1.4$ in all the Riemann problems presented here. In the 1D tests the initial discontinuity is located at $x = 0.5$.

Contact Discontinuity A 1D Riemann problem involving an isolated contact discontinuity illustrates the advantage of using the the HLLC Riemann solver as opposed to the HLL Riemann solver. Here we present results from the stationary and slowly moving contact discontinuity tests (cf. Toro 2009; Liska and Wendroff 2003) with initial conditions (left and right states)

$$\begin{aligned} [\rho,\, v,\, p]_L &= [1.4,\, v_{\rm c},\, 1.0] \\ [\rho,\, v,\, p]_R &= [1.0,\, v_{\rm c},\, 1.0]\,, \end{aligned} \tag{46}$$

where the speed of the contact discontinuity is $v_{\rm c} = 0.0$ and $v_{\rm c} = 0.1$ for the stationary and the slowly moving contact discontinuity, respectively. The system is evolved until $t = 2.0$, at which time the moving contact discontinuity is located at $x = 0.7$.

Results from the stationary and slowly moving contact discontinuity tests for $t = 2$ are plotted in Figure 8 (left and right panel, respectively). These tests clearly demonstrate the improved resolution of the contact discontinuity when the HLLC Riemann solver is used. The HLLC solver is exact for the stationary contact discontinuity. This is because the middle wave speed estimate given by Equation (34) is exact in this case, and results in zero mass flux across the discontinuity. The diffusive part of the HLL flux (cf. the third term on the right-hand side of Equation (17)) results in a non-zero mass flux across the contact discontinuity, even as v_x remains zero. Both solvers are inexact for the moving contact discontinuity test, but the HLLC solver remains superior.

Sod Shock Tube The Sod shock tube is a well-known Riemann problem with an analytic solution. It was introduced by Sod (1978) to benchmark algorithms for solving the Euler equations. The problem is initialized with left and right states given by

$$\begin{aligned} [\rho,\, v,\, P]_L &= [1.0,\, 0.0,\, 1.0]\,, \\ [\rho,\, v,\, P]_R &= [0.1,\, 0.0,\, 0.125]\,. \end{aligned} \tag{47}$$

For $t > 0$, nonlinear waves are generated and propagate away from the initial discontinuity. A shock wave propagates to the right and a rarefaction wave propagates to the left. Also, a contact discontinuity propagates to the right, between the shock and the rarefaction waves.

Figure 9 shows results from this test problem for $t = 0.25$. GenASiS captures all the essential features of this Riemann problem with good accuracy. In particular, we have compared the numerical results with the analytic solution using the L_1 error norm: Table 4 shows the L_1 error norm and the convergence rate (for mass density and pressure) obtained with both the HLL and the HLLC Riemann solvers. The errors obtained when using the HLLC Riemann solver are slightly smaller, especially in the mass density. This is because the HLLC solver better resolves the contact discontinuity, across which only the density varies. However, both solvers result in similar first-order convergence rate, which is expected for problems involving discontinuities. (We also ran this problem in two- and three-dimensional mode by letting the waves propagate along the y- and z-coordinate directions, and the results from those runs are exactly the same as the results listed in Table 4.)

We have also used a three-dimensional version of the Sod shock tube to study the parallel scaling behavior of the hydrodynamics solver in GenASiS (cf. Algorithm 1). Figure 10 shows results from a pure MPI weak-scaling test on Jaguar, the Cray XT-5 machine at the Oak Ridge Leadership Computing Facility (OLCF). The hydrodynamics algorithms implemented in GenASiS scale well up to about 10^5 MPI tasks.

Double Rarefaction The double rarefaction problem was introduced by Einfeldt et al. (1991) to test the behavior of Riemann solvers on problems where a 'vacuum' is created in a region between two strong rarefaction waves propagat-

ing away from an initial discontinuity in the velocity v_x. In particular, the problem was designed to reveal the lack of positivity conservation (e.g. resulting in negative pressure) by Riemann solvers based on linearization (e.g. the solver provided by Roe 1981), while showing that the HLLE solver (which is similar to the HLL scheme in Equation 17) is positivity conserving (in 1D), provided that certain constraints on the wave speeds $\alpha^x_\pm$ are satisfied.

We initialize the double rarefaction problem as described by Einfeldt et al. (1991) (problem 1-2-0-3; see also Toro 2009). The left and right states of the Riemann problem are given by

$$\begin{aligned} [\rho,\, v,\, p]_L &= [1.0,\, -2.0,\, 0.4] \\ [\rho,\, v,\, p]_R &= [1.0,\, +2.0,\, 0.4]\,. \end{aligned} \tag{48}$$

Figure 11 shows results from running the double rarefaction problem with GenASiS to time $t = 0.15$, using 128 cells and the HLLC Riemann solver. A referential solution using 1000 cells is plotted with a solid black line. The referential solution was obtained with an exact Riemann solver program (`e1rpexf.f`) from the NUMERICA library (Toro 1999, see also the discussion in Chapter 4 in Toro 2009).

The result shows that GenASiS follows the solution obtained with the exact Riemann solver well. Moreover, the density and pressure remain positive. However, the specific internal energy e/ρ shows a pathology around the location of the initial discontinuity $x = 0.5$. (Both the density and pressure approach zero, while the specific internal energy remains finite in this problem.) This pathology is likely due to our use of the conservative formulation of the Euler equations (i.e., we evolve the total energy density) in a situation where the kinetic energy density is significantly larger (a factor of two initially) than the internal energy density (see also results from this test (test 2) in Liska and Wendroff 2003, obtained with multiple Eulerian schemes for hydrodynamics), which can result in an inaccurate internal energy density when it is obtained by subtracting the kinetic energy density from the total (or balanced) energy density G (cf. Equation (31); Blondin and Lufkin 1993). For the specific internal energy, we also plot results from runs using 256 cells (blue dashed line) and 512 cells (green dotted line). The results converge to the exact solution, albeit very slowly near the center.

Implosion This is a 2D Riemann problem with initial conditions similar to the Sod shock tube. It was used by Liska and Wendroff (2003) to compare several numerical schemes for solving the Euler equations (see also Stone et al. 2008). The problem is solved on a square computational domain confined to $[x, y] \in [0, 0.3] \times [0, 0.3]$, with reflecting boundary conditions everywhere. The fluid is initially at rest, and the density and pressure are set to $\rho = 0.125$ and $p = 0.14$ in the region where $x + y \leq 0.15$, and to $\rho = 1$ and $p = 1$ elsewhere.

Results obtained with GenASiS using 400×400 cells and the HLLC Riemann solver are shown for select times ($t = 0.045$ and $t = 2.5$) in Figure 12, which is a color map of the pressure, with density contours and velocity vectors overlaid. (Figure 12 can be compared directly with Figures 4.10 and 4.11 in Liska and Wendroff (2003), and Figure 17 in Stone et al. (2008).) At $t = 0.045$ a shock propagates diagonally towards the origin, and a rarefaction wave propagates in the opposite direction. A contact discontinuity is trailing the shock (cf. the density contours in the left panel in Figure 12). The evolution along the diagonal is similar to that of the Sod shock tube for $t = 0.045$. At later times, wave-boundary and wave-wave interactions eventually result in a very complex flow structure.

The results obtained with GenASiS compare favorably with results obtained with other dimensionally unsplit codes (e.g. Liska and Wendroff 2003; Stone et al. 2008). In particular, the symmetry about the diagonal connecting $(0, 0)$ and $(0.3, 0.3)$ is perfectly preserved. There is no analytical solution to the implosion problem. However, Stone et al. (2008) argue that the production of a jet along the diagonal is part of the correct result for this test. The jet along the diagonal is clearly seen in the density contours in the right panel of Figure 12. The formation and subsequent evolution of the jet is very sensitive to the numerical scheme's ability to preserve the initial symmetry. We also find

the formation and evolution of the jet to be sensitive to the scheme's ability to track the intermediate waves: the jet is absent when the HLL Riemann solver in GenASiS is used, and the solution then resembles the results produced with the Positive scheme (LL) in Liska and Wendroff (2003). This is not surprising, as the jet is formed from vortices produced near the origin, which are later advected with the fluid velocity along the diagonal (Stone et al. 2008).

4.2.2. *InteractingBlastWaves*

This problem, discussed in detail by Woodward and Colella (1984), involves multiple interactions between shocks, rarefaction waves, and contact discontinuities. It is considered an extremely difficult test for methods employing a uniform Eulerian mesh (Woodward and Colella 1984), and has been used by many authors to benchmark solvers for the Euler equations (e.g. Kurganov et al. 2001; Liska and Wendroff 2003; Stone et al. 2008). The problem is initialized on a computational domain confined to $x \in [0, 1]$ with reflecting boundary conditions. The fluid is initially at rest with constant background density $\rho_0 = 1$ and pressure $p_0 = 0.01$. Two initial pressure jumps are introduced by setting the pressure to $p = 1000$ in the region where $x < 0.1$, and to $p = 100$ in the region where $x > 0.9$. For $t > 0$, strong shocks, rarefactions, and contact discontinuities develop as a result of the initial pressure jumps, which later interact multiple times to create a complex flow structure.

Figure 13 shows results obtained with GenASiS for $t = 0.038$, using 400 cells and the HLLC Riemann solver. A high-resolution reference solution, obtained by using 10^4 cells, is also included in the plot (solid black line). The results obtained with GenASiS are comparable to those obtained by other authors (cf. Figures 4.7 and 4.9 in Kurganov et al. 2001, Figure 3.10 in Liska and Wendroff 2003, and Figure 9 in Stone et al. 2008). For $t = 0.038$, the maximum density is about 5.4, which is significantly lower than in the reference solution (6.4), but comparable to the results presented by Kurganov et al. (2001), and the results obtained with many of the schemes tested by Liska and Wendroff (2003). However, our maximum density is somewhat lower than the value obtained by Stone et al. (2008), who used third-order spatial reconstruction. Moreover, the contact discontinuity located around $x = 0.6$ is poorly resolved, but the results obtained with GenASiS are comparable to results obtained with other schemes based on a fixed Eulerian mesh (e.g. Kurganov et al. 2001; Liska and Wendroff 2003; Stone et al. 2008). Schemes based on a moving (e.g. Lagrangian) meshes perform very well on this test (e.g. Woodward and Colella 1984; Springel 2010).

4.2.3. *SedovTaylorBlastWave*

The Sedov-Taylor blast wave is a classic test in computational astrophysics. It has been used by many authors to benchmark multidimensional hydrodynamics algorithms (e.g. Fryxell et al. 2000; Almgren et al. 2010; Springel 2010; Käppeli et al. 2011). It follows the self-similar evolution of a strong shock wave expanding into a uniform medium. We follow closely the problem setup described in Fryxell et al. (2000), and we present results from 2D (cylindrical detonation) and 3D (spherical detonation) computations, employing both a single-level and a fixed multilevel grid. The problem is initialized with a fluid at rest, with uniform density $\rho_0 = 1$ and (small) pressure $p_0 = 10^{-5}$. The computational domain is confined to $[-0.5, 0.5]$ in each coordinate dimension in these runs. The adiabatic index is set to $\gamma = 1.4$. An amount of thermal energy $E_d = 1$ is instantaneously released inside a finite detonation radius, R_d, resulting in a pressure

$$p_d = \frac{3(\gamma - 1)E_d}{(\alpha + 1)\pi R_d^\alpha} \tag{49}$$

in the detonation region $r \leq R_d$, where $\alpha = 2$ and $r = \sqrt{x^2 + y^2}$ for the 2D version of the test, and $\alpha = 3$ and $r = \sqrt{x^2 + y^2 + z^2}$ for the 3D version. For $t > 0$, the detonation results in the formation and expansion of a

strong cylindrical (2D) or spherical (3D) shock wave. From dimensional arguments, the shock radius is approximately $R_{\rm Sh}(t) \approx \left(E_d t^2/\rho_0\right)^{1/(\alpha+2)}$, and the velocity of the expanding shock wave is $\dot{R}_{\rm Sh} \approx 2(\alpha+2)^{-1}\,(R_{\rm Sh}/t)$. At $t = 0.05$ the shock has reached $r \approx 0.224$ in the 2D version of the test and $r \approx 0.302$ in the 3D version. From the shock jump conditions in Equation (13) we find the following values for density, flow velocity, and pressure immediately behind the shock for $t = 0.05$: $\rho_{\rm Sh}^- \approx 6.0$, $v_{\rm Sh} \approx 1.867$, and $p_{\rm Sh}^- \approx 4.181$ (2D), and $\rho_{\rm Sh}^- \approx 6.0$, $v_{\rm Sh} \approx 2.013$, and $p_{\rm Sh}^- \approx 4.864$ (3D).

We find that the outcome of this test—in particular, the final shock position—is sensitive to the way the detonation is initiated. Ideally, the detonation occurs in a single point. However, for practical computations with a finite volume scheme employing an Eulerian Cartesian mesh, the detonation radius is typically spread out over three cells (Fryxell et al. 2000). To further improve the initialization of the blast wave, we subdivide each cell that is intersected by the surface of the sphere with radius R_d into a subgrid with 20 cells in each coordinate direction (Almgren et al. 2010). The pressure in the intersected cells is then obtained from a volume average over the subgrid.

Results from 2D and 3D single-level mesh calculations using the HLL Riemann solver are displayed in Figures 14 and 15. (Results obtained with the HLL and the HLLC Riemann solvers are nearly identical for this test, which primarily involves an expanding strong shock, so that the contact wave captured by the HLLC solver plays a minimal role.) Scatter plots of the density, the velocity magnitude, and the pressure versus radius, obtained by using 256 cells per dimension, are displayed in the two upper panels and in the lower left panel in each figure, respectively. Despite the fact that our results are obtained by using a single-level mesh, they compare reasonably well with the results presented in Fryxell et al. (2000), which were obtained with an adaptively refinable mesh. Most noticeably, the peak values of the density and pressure just behind the shock are somewhat lower than the analytic solution, but the agreement improves with increasing spatial resolution (cf. lower right panel in Figure 14). The shock position (relative to the analytic solution) also improves with increasing spatial resolution. The shock position in the 3D run overshoots the analytic value by about 3% at $t = 0.05$, but the agreement between the analytic and numerical results seems to improve for later times (cf. Figure 18). The color plot of the pressure in the xy-plane ($z = 0$) from the 3D run with 256^3 cells (lower right panel in Figure 15) illustrates the spherical shape of the shock surface at $t = 0.05$; i.e., 'grid imprints' in the shape of the shock, due to our use of Cartesian coordinates, are minimal.

Figures 16-18 display results from the Sedov-Taylor blast wave test, obtained with the fixed multilevel grid capabilities in GenASiS. The 2D results (employing seven mesh levels) are shown in Figures 16 and 17, and the 3D results (employing four mesh levels) are shown in Figure 18. For both tests, the multilevel mesh results are compared with corresponding single-level mesh results. The multilevel meshes consist of concentric nested spheres (embedded in a square or cubic box at the coarsest level) with increasing resolution towards the origin. The spatial resolution of adjacent levels differs by a factor of two. This type of mesh is well suited for problems with a centrally condensed matter distribution, and we intend to employ a similar multilevel mesh in future initial simulations of core-collapse supernovae with GenASiS. (The current multilevel mesh capabilities in GenASiS only allow for evolution on a static mesh, but we intend to implement adaptive mesh refinement capabilities in the future.) For a 'fair' comparison, the single-level mesh results are obtained with a resolution equal to the resolution of the deepest level in the multilevel mesh (i.e., 1024^2 cells in 2D and 128^3 cells in 3D). Of course, the multilevel mesh results become poorly resolved after the shock has crossed multiple fine-to-coarse mesh boundaries. The single-level mesh results are only included to show that the multilevel mesh results look reasonable in comparison (e.g. the shock positions are similar). Besides showing reasonable comparison with the single-level mesh results, the main purpose of including these tests is to demonstrate that the multilevel time integration algorithm, with subcycling of deeper levels and conservative flux updates across coarse-fine mesh boundaries (cf. Algorithms 2-4), work as intended.

The shock has crossed one level boundary in the 2D run displayed in the left panel in Figure 16. The shock radius is very similar in the two runs. (In the analytic solution, $R_{\rm Sh} \approx 0.1$.) The maximum pressure (i.e., just behind the

shock) is ~ 17 in the multilevel mesh run, and ~ 18.5 in the single-level mesh run (versus $p_{\rm Sh}^{-} \approx 20.8$ in the analytic solution). Later ($t = 0.04$), when the shock has crossed three fine-to-coarse level boundaries, the shock positions are still reasonably similar in the two runs (considering the factor of 8 difference in spatial resolution), with $R_{\rm Sh} \sim 0.2$ ($R_{\rm Sh} \approx 0.2$ in the analytic solution). Just behind the shock, the pressure is ~ 3.7 in the multilevel mesh run, and about 4.8 in the single-level mesh run ($p_{\rm Sh}^{-} \approx 5.2$ in the analytic solution for $t = 0.04$). The comparison between the multilevel and single-level mesh 3D runs in Figure 18 also indicates reasonable agreement (considering that the shock in the multilevel mesh run has reached level 2, where the effective resolution is only 32^3, for $t = 0.1$). Moreover, the total mass, linear momentum, and energy in the multilevel mesh runs are conserved to numerical precision.

4.3. FluidInstabilityTests

The `FluidInstabilityTests` division of `Applications` (see the middle left diagram in Figure 6) includes the driver programs `KelvinHelmholtz` and `RayleighTaylor`, which demonstrate how two instabilities of astrophysical relevance grow and develop in GenASiS simulations.

4.3.1. KelvinHelmholtz

The Kelvin-Helmholtz (KH) instability is relevant to a broad range of astrophysical applications. For example, the shear flows induced by the spiral mode of the so-called standing accretion shock instability (SASI; Blondin et al. 2003) are KH unstable. Energy transfer, mediated by the KH instability, from flows associated with "low-order" SASI modes to small-scale turbulent flows, can possibly result in the nonlinear saturation of the SASI (Guilet et al. 2010; Endeve et al. 2012). We include a 2D version of the KH test here to further highlight differences between simulation results obtained when using the HLL and HLLC Riemann solvers in GenASiS, and to compare with (and hopefully corroborate) findings reported by other authors. In particular, simulations of the relativistic magnetohydrodynamic KH instability (e.g. Mignone et al. 2009; Beckwith and Stone 2011) have revealed that schemes based on approximate Riemann solvers that include the intermediate waves in the Riemann fan have significantly improved spectral resolution when compared to schemes that do not. Note that the inclusion of this test is *not* an attempt to study any aspect of the KH instability.

Our numerical setup of the KH instability test follows closely the description detailed on the Athena website.[4] The computational domain is confined to $[x, y] \in [0, 1] \times [0, 1]$, with periodic boundary conditions in both spatial dimensions. We denote the lengths of the x and y sides of the computational domain with L_x and L_y respectively. Velocity shear layers are located at $y = 0.25$ and $y = 0.75$. The pressure is initially uniform everywhere, with $p = 2.5$. In the region with $|y - 0.5| < 0.25$ we set $\rho = 2$ and $v_x = 0.5$, while in the region of the computational domain where $|y - 0.5| \geq 0.25$ we set $\rho = 1$ and $v_x = -0.5$. For the unperturbed initial state we set $v_y = v_z = 0$ everywhere. The adiabatic index is set to $\gamma = 1.4$.

The shear layers in the initial configuration are KH unstable to perturbations in the velocity components perpendicular to the initial flow (e.g. v_y; Chandrasekhar 1981). For our initial setup, the growth rate of a single-mode perturbation with associated wavenumber k_x is (cf. Section 101 in Chandrasekhar 1981)

$$\Gamma_{\rm KH} = k_x \frac{\sqrt{2}}{3} \Delta_y v_x, \tag{50}$$

[4] www.astro.princeton.edu/∼jstone/Athena/tests/kh/kh.html

where $\Delta_y v_x$ is the jump in v_x across the velocity shear layer. Thus the amplitude of a single mode perturbation with associated wavelength λ_x grows exponentially with growth time $\Gamma_{\rm KH}^{-1} \approx 0.34\,(\lambda_x/L_x)$; perturbations associated with shorter wavelengths grow at a faster rate. We initiate the KH instability with perturbations in the initial velocity field $\mathbf{v}_0$ by setting

$$\mathbf{v} = \mathbf{v}_0 + \delta\mathbf{v}, \tag{51}$$

where we use a combination of single-mode and random-mode perturbations

$$\delta\mathbf{v} = [A_{\rm s} \sin\,(2\pi x/L_x) + A_{\rm R}] \times \psi(y)\,\hat{\mathbf{y}}. \tag{52}$$

The amplitude of the single-mode perturbation is $A_{\rm s} = 10^{-2}$, and $A_{\rm R}(x, y)$ assumes random numbers between -10^{-3} and 10^{-3}. The function $\psi(y)$ concentrates the perturbations in the shear layers

$$\psi(y) = \exp\left[-\frac{1}{2}\left(\frac{\cos\,(2\pi y/L_y)}{2\pi\sigma/L_y}\right)^2\right], \tag{53}$$

where we set $\sigma = 0.1$. The random perturbations seed small-scale modes, whose growth may be suppressed by an excessively dissipative numerical scheme.

We have carried out simulations using 256^2 (low resolution) and 512^2 (high resolution) cells, using both the HLL and HLLC Riemann solvers in GenASiS. The simulations are evolved until $t = 5.0$. We display the density distribution at select times in Figures 19-21 ($t = 0.6$, Figure 19; $t = 1.0$, Figure 20; $t = 2.0$, Figure 21), which illustrate the results from the high resolution runs. (Results obtained with the HLL and HLLC solvers are displayed in the left and right panels, respectively.) The differences in the results obtained with the two Riemann solvers become apparent at an early stage. The run with the HLLC solver has developed small-scale 'KH rolls' in the shear layer at $t = 0.6$. These are absent in the run with the HLL solver, which seems to have only been affected by the sinusoidal part of the perturbation at this point. The interfaces between the low and high density fluids are clearly more diffuse in the HLL run, which is also to be expected, as this solver ignores the contact and shear waves in the Riemann fan. Small-scale KH rolls, although clearly affected by the more diffuse interface, have developed in the HLL run for $t = 1.0$. The shear layers in the HLLC run are much more disturbed by the instability at this time. The two runs share some qualitative similarities at $t = 2.0$ (i.e., the largest-scale component of the dense-fluid deformation), but detailed visual comparisons become increasingly difficult with evolving time. The shear layers in the HLL run are clearly populated with KH rolls of roughly equal size at $t = 2.0$, while the deformations of the same layers in the HLLC run can be characterized as covering a broader spectrum of spatial scales.

Figure 22 provides additional quantitative results from the KH runs. In the left panel we plot the y-component of kinetic energy,

$$E_{\rm kin,y} = \int_V \frac{1}{2}\rho v_y^2\, dV, \tag{54}$$

versus time for the two high-resolution runs. The grey line is from the HLL run, while the black line is from the HLLC run. Note that $E_{\rm kin,y}$ grows faster initially in the HLLC run than in the HLL run. This growth is due to the developments seeded by the small-scale random perturbations. In both runs, $E_{\rm kin,y}$ reach similar levels at late times ($t \gtrsim 2.0$), but the distribution of kinetic energy on various spatial scales remains different. In the right panel of Figure 22 we plot the spectral distribution of the kinetic energy $\hat{e}_{\rm kin}(k)$ at $t = 2.0$ from the low (dashed lines) and high (solid lines) resolution runs. (The wavenumber, $k = 2\pi/\lambda_k$, is the magnitude of the wavevector $\mathbf{k}$.) The energy spectra are obtained from Fourier transforms of the components of $\sqrt{\rho}\mathbf{v}$, and satisfy

$$\int_0^\infty \hat{e}_{\rm kin}(k)\, dk = \int_V \frac{1}{2}\rho|\mathbf{v}|^2\, dV \tag{55}$$

(cf. Ryu et al. 2000). Clearly, the HLLC solver results in higher spectral resolution for a given spatial resolution. The spectra from the high-resolution runs begin to separate already around $k = 40$ (i.e., $\lambda_k \approx 80\,\Delta x$), and the spectrum from the HLL run falls below 10^{-9} around $k \approx 560$, while the spectrum from the HLLC run stays above 10^{-9} out to $k \approx 920$. Moreover, the spectrum from the low-resolution run with the HLLC solver follows very closely the spectrum from the high-resolution run with the HLL solver. Thus the low-resolution run with the HLLC solver offers the same spectral resolution as the high-resolution HLL run. This is a potentially substantial computational savings, especially for 3D simulations.

In general, the conclusions we can draw from these runs agree very well with those of other authors (e.g. Mignone et al. 2009; Beckwith and Stone 2011). The inclusion of the intermediate waves in the approximate Riemann solver results in sharper contact and shear discontinuities, and unphysical suppression of the growth of KH unstable modes may be avoided, which also impacts the overall evolution of the instability—in particular its growth rate. Significantly improved spectral resolution of the nonlinear flows is also obtained when the HLLC solver is employed in GenASiS.

4.3.2. RayleighTaylor

The Rayleigh-Taylor (RT) instability is another important fluid instability with relevance to astrophysical applications. The RT instability is extensively analyzed in Chandrasekhar (1981). Again, we follow closely the initial setup described on the Athena website (see also Liska and Wendroff 2003, who compared results from this test computed with eight different schemes). The 2D computational domain is confined to $[x, y] \in [-0.25, 0.25] \times [-0.75, 0.75]$, and we use periodic boundary conditions at $|x| = 0.25$ and reflecting boundary conditions at $|y = 0.75|$. This test involves a uniform external force, and is the only test with nonzero sources included in this paper. The problem consists of a denser fluid above a less dense fluid at rest, with $\rho = \rho_2 = 2$ for $y > 0$ and $\rho = \rho_1 = 1$ for $y \leq 0$. The external force acts anti-parallel to $\hat{\mathbf{y}}$ with a constant acceleration $g = 0.1$. Setting $\Phi(y_l) = 0$, the corresponding potential is $\Phi = g(y - y_l)$. The pressure is $p = p_b - \rho g y$ with $p_b = 2.5$. Thus the initial configuration is in hydrostatic equilibrium $\nabla p + \rho \nabla \Phi = 0$. The adiabatic index is set to $\gamma = 1.4$.

We specify the potential in the cell centers and compute its gradient using second-order finite differences. Because the analytic potential is linear in y, the computation of its gradient is exact. In particular,

$$\langle \mathcal{S} \rangle = [0, 0, m\langle D \rangle, 0, \langle S^y \rangle]^T \times \left(\frac{\Phi_{\leftrightarrow y+} - \Phi_{-y\leftrightarrow}}{y_{\leftrightarrow y+} - y_{-y\leftrightarrow}} \right). \tag{56}$$

are the source terms in Equation (2)—the external force and power.

In the absence of surface tension at the interface between the two fluids, the initial configuration described above is unstable to perturbations in all wavenumbers $k_x = 2\pi/\lambda_x$. The growth rate of a single-mode perturbation is (cf. §92 Chandrasekhar 1981)

$$\Gamma_{\text{RT}} = \sqrt{g k_x \mathrm{A}}, \tag{57}$$

where $\mathrm{A} = (\rho_2 - \rho_1)/(\rho_2 + \rho_2)$ is the Atwood number, which is $1/3$ for our initial setup. Thus the growth time for a single-mode perturbation with wavelength λ_x is $\Gamma_{\text{RT}}^{-1} \approx 1.55\,(\lambda_x/L_x)^{-1/2}$.

We initiate the RT instability by perturbing the initial velocity field $\mathbf{v} = \mathbf{v}_0 + \delta\mathbf{v}$, with $\mathbf{v}_0 = 0$ and a single-mode perturbation of the form

$$\delta\mathbf{v} = \frac{A_{\text{s}}}{4} \left[1 + \cos(2\pi x/L_x)\right] \times \left[1 + \cos(3\pi y)\right] \hat{\mathbf{y}}, \tag{58}$$

with amplitude $A_{\text{s}} = 10^{-2}$.

Figure 23 displays results from the RT instability test computed with 256×768 cells. The left panel shows the density distribution at $t = 8.5$ from a simulation using the HLLC Riemann solver. The instability has evolved well into the nonlinear stage at this time, with the characteristic rising bubble (or 'mushroom cap') and falling spikes clearly displayed. On the top side of the bubble there are signs of emerging secondary KH instabilities at the interface between the dense and light fluids. In the right panel we plot contours of constant density $\rho = 1.25$, comparing results obtained with the HLL (solid black) and the HLLC (dashed grey) Riemann solvers. The results computed with the two Riemann solvers agree qualitatively, but again the HLLC solver preserves a sharper interface between the fluids. This becomes particularly evident when looking at the 'coil-up' of the two fluids below the mushroom cap.

5. CONCLUSION

This paper is the second in a series on GenASiS (*Gen*eral *A*strophysical *Si*mulation *S*ystem), a new code ultimately aimed at state-of-the-art simulations of core-collapse supernovae and other astrophysical problems. Its design facilitates reference to multiple algorithms, solvers, and physics and numerics choices with the same abstracted names and/or interfaces, through use of Fortran 2003 features that support the object-oriented programming paradigm (e.g. Reid 2007; Adams et al. 2008). As seen in Figure 1, the three major divisions of GenASiS are `Basics`, which contains some basic utilitarian functionality for large-scale simulations on distributed-memory supercomputers; `Mathematics`, which includes generic mathematical constructs and solvers that are as agnostic as possible with regard to the specifics of any particular system; and `Physics`, which sets up physical spaces associated with various theories of spacetime (including gravity), defines various forms of stress-energy, and combines these into 'universes.' Paper I covered `Basics` and one of the divisions of `Mathematics`: cell-by-cell refinable `Manifolds`.

By way of documenting the algorithms we have implemented for nonrelativistic hydrodynamics, in this second paper we introduce `Solvers`—another division of `Mathematics`—and the `Physics` division of GenASiS. `Solvers` (see Figure 2) includes finite-volume updates for generic hyperbolic `BalanceEquations`, using second-order spatial `Reconstructions` with capabilities for the HLL and HLLC approximate `RiemannSolvers`; and ordinary differential equation integration `Steps`, in this case a second-order Runge-Kutta time stepper. `Physics` (see Figure 5) extends the `Manifolds` division of `Mathematics` into physical `Spaces`, in this case a nonrelativistic gravity-free `Galilean` space; defines `StressEnergies`, in this case pressureless dust and polytropic `Fluids`; and combines classes from `Spaces` and `StressEnergies` into `Universes`, in this case a `FreeFluidForm` whose evolution draws on the `BalanceEquations` division of `Solvers`.

We also present our first `Applications` in the form of several standard test problems against which we benchmark the hydrodynamics capabilities of GenASiS. Included (see Figure 6) are `SmoothFluidTests` that enable us to determine the order of accuracy of the hydrodynamics solvers; `DiscontinuousFluidTests` that test the ability of GenASiS to handle shocks and other discontinuities; and `FluidInstabilityTests` that demonstrate how two instabilities of astrophysical relevance grow and develop in GenASiS simulations. These tests illustrate the basic competence of the classes relevant to nonrelativistic hydrodynamics; demonstrate second-order convergence for problems with smooth analytic solutions, and first-order convergence for problems with discontinuous solutions; and produce results that are comparable to and consistent with those presented by other authors. The superiority of the HLLC Riemann solver over the HLL Riemann solver is especially apparent in the contact discontinuity test (Figure 8) and tests of the Kelvin-Helmholtz (Figures 19-22) and Rayleigh-Taylor (Figure 23) instabilities. Moreover, we have provided preliminary indications of the code's ability to scale to a large number of MPI tasks (Figure 10), and demonstrated the functionality of our explicit multilevel time stepping algorithm for the evolution of balance equations with cell-by-cell fixed-mesh refinement (Figures 16–18). As we continue the development of GenASiS, the test problems presented here will form the basis for a comprehensive regression test suite in Bellerophon (Lingerfelt et al. 2011), in

order to ensure continued reliable performance and guard against the unintentional introduction of code bugs.

Subsequent papers in this series will document additions to `Mathematics` and `Physics` that will make GenASiS suitable for multiphysics simulations of core-collapse supernovae and other astrophysical problems. Development of additional solvers and physics are already underway. A nuclear equation of state is one obvious requirement. We have implemented magnetohydrodynamics (MHD) capabilities in a 'draft version' of GenASiS (Endeve et al. 2012), which has been used to study standing accretion shock instability (SASI)-driven magnetic field amplification (Endeve et al. 2010, 2012). In the current version of GenASiS we plan to implement MHD capabilities based on the HLLD solver (Miyoshi and Kusano 2005; Mignone et al. 2009). For Newtonian gravity, a multilevel Poisson solver will use a distributed FFT solver (Budiardja and Cardall 2011) for the coarsest level, in conjunction with finite-difference solves on individual levels, in a multigrid approach. Work on general relativistic gravity—a major undertaking—is also underway in GenASiS (Tsatsin et al. 2011). The greatest challenge of all is neutrino transport. Building on our group's past experience (Liebendörfer et al. 2004; Bruenn et al. 2009), theoretical developments (Cardall and Mezzacappa 2003; Cardall et al. 2005), and initial forays into transport in the earliest version of GenASiS (Cardall et al. 2006; Cardall 2009), we plan to deploy a multigrid approach here as well, using a multigroup Variable Eddington Tensor formulation (Endeve et al. 2012, in preparation) with closures of increasing sophistication, ultimately culminating in a full 'Boltzmann solver.'

This research was supported by the Office of Advanced Scientific Computing Research and the Office of Nuclear Physics, U.S. Department of Energy. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory provided through the INCITE program.

REFERENCES

Adams, J. C., Brainerd, W. S., Hendrickson, R. A., Maine, R. E., Martin, J. T., and Smith, B. T. (2008). *The Fortran 2003 Handbook: The Complete Syntax, Features and Procedures*. Springer. 2, 26

Almgren, A. S., Beckner, V. E., Bell, J. B., Day, M. S., Howell, L. H., Joggerst, C. C., Lijewski, M. J., Nonaka, A., Singer, M., and Zingale, M. (2010). CASTRO: A New Compressible Astrophysical Solver. I. Hydrodynamics and Self-Gravity. *The Astrophysical Journal*, 715(2):1221–1238. 21, 22

Batten, P., Clarke, N., Lambert, C., and Causon, D. M. (1997). On the Choice of Wavespeeds for the HLLC Riemann Solver. *SIAM Journal on Scientific Computing*, 18(6):1553. 2, 6, 12

Beckwith, K. and Stone, J. M. (2011). a Second-Order Godunov Method for Multi-Dimensional Relativistic Magnetohydrodynamics. *The Astrophysical Journal Supplement Series*, 193(1):6. 23, 25

Blondin, J. M. and Lufkin, E. A. (1993). The Piecewise-Parabolic Method in Curvilinear Coordinates. *The Astrophysical Journal Supplement Series*, 88:589–594. 20

Blondin, J. M., Mezzacappa, A., and DeMarino, C. (2003). Stability of Standing Accretion Shocks, with an Eye toward CoreCollapse Supernovae. *The Astrophysical Journal*, 584(2):971–980. 23

Bruenn, S. W., Mezzacappa, A., Hix, W. R., Blondin, J. M., Marronetti, P., Messer, O. E. B., Dirk, C. J., and Yoshida, S. (2009). 2D and 3D core-collapse supernovae simulation results obtained with the CHIMERA code. *Journal of Physics: Conference Series*, 180:012018. 27

Budiardja, R. D. and Cardall, C. Y. (2011). Parallel FFT-based Poisson solver for isolated three-dimensional systems. *Computer Physics Communications*, 182(10):2265–2275. 27

Cardall, C., Budiardja, R., Endeve, E., and Mezzacappa, A. (2012). GenASiS: *Gen*eral *A*strophysical *Si*mulation *S*ystem. I. Fundamentals. *ApJS, submitted*. 2, 3, 8, 10, 11, 13, 14, 26, 34

Cardall, C., Lentz, E., and Mezzacappa, A. (2005). Conservative special relativistic radiative transfer for multidimensional astrophysical simulations: Motivation and elaboration. *Physical Review D*, 72(4). 27

Cardall, C. and Mezzacappa, A. (2003). Conservative formulations of general relativistic kinetic theory. *Physical Review D*, 68(2). 27

Cardall, C. Y. (2009). An approach to neutrino radiative transfer in supernova simulations. In Kanschat, G., Meinköhn, E., Rannacher, R., and Wehrse, R., editors, *Numerical Methods in Multidimensional Radiative Transfer*, pages 27–37, Berlin Heidelberg. Springer. 27

Cardall, C. Y., Razoumov, A. O., Endeve, E., and Mezzacappa, A. (2006). The Long Term: Six-dimensional Core-collapse Supernova Models. In Mezzacappa, A. and Fuller, G. M., editors, *Open Issues in Core Collapse Supernova Theory*. World Scientific Publishing Company, London, England. 27

Chandrasekhar, S. (1981). *Hydrodynamic and Hydromagnetic Stability (International Series of Monographs on Physics)*. Dover Publications. 23, 25

Davis, S. F. (1988). Simplified Second-Order Godunov-Type Methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445. 6

Del Zanna, L. and Bucciantini, N. (2002). An efficient shock-capturing central-type scheme for multidimensional relativistic flows I. Hydrodynamics. *Astronomy and Astrophysics*, 390(3):1177–1186. 2, 6

Del Zanna, L., Bucciantini, N., and Londrillo, P. (2003). An efficient shock-capturing central-type scheme for multidimensional relativistic flows II. Magnetohydrodynamics. *Astronomy and Astrophysics*, 400(2):397–413. 2, 6

Del Zanna, L., Zanotti, O., Bucciantini, N., and Londrillo, P. (2007). ECHO: a Eulerian conservative high-order scheme for general relativistic magnetohydrodynamics and magnetodynamics. *Astronomy and Astrophysics*, 473(1):11–30. 2, 6

Duez, M., Liu, Y., Shapiro, S., and Stephens, B. (2005). Relativistic magnetohydrodynamics in dynamical spacetimes: Numerical methods and tests. *Physical Review D*, 72:024028. 2, 6

Einfeldt, B. (1988). On Godunov-Type Methods for Gas Dynamics. *SIAM Journal on Numerical Analysis*, 25(2):294. 2, 6

Einfeldt, B., Munz, C., Roe, P., and Sjögreen, B. (1991). On Godunov-type methods near low densities. *Journal of Computational Physics*, 92(2):273–295. 19, 20

Endeve, E., Cardall, C. Y., Budiardja, R. D., Beck, S. W., Bejnood, A., Toedte, R. R., Mezzacappa, A., and Blondin, John, M. (2012). Turbulent Magnetic Field Amplification from Spiral SASI Modes: Implications for Core-Collapse Supernovae and Proto-Neutron Star Magnetization. *The Astrophysical Journal, in press (arXiv:1203.3108)*. 23, 27

Endeve, E., Cardall, C. Y., Budiardja, R. D., and Mezzacappa, A. (2010). Generation of Magnetic Fields by The Stationary Accretion Shock Instability. *The Astrophysical Journal*, 713(2):1219–1243. 27

Endeve, E., Y Cardall, C., Budiardja, R., and Mezzacappa, A. (2012). Turbulent magnetic field amplification from spiral SASI modes in core-collapse supernovae. *Journal of Physics: Conference Series, in press (arXiv:1203.3385)*. 27

Farris, B., Li, T., Liu, Y., and Shapiro, S. (2008). Relativistic radiation magnetohydrodynamics in dynamical spacetimes: Numerical methods and tests. *Physical Review D*, 78(2):1–20. 6

Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W., and Tufo, H. (2000). FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *The Astrophysical Journal Supplement Series*, 131(1):273–334. 16, 21, 22

Gammie, C. F., McKinney, J. C., and Toth, G. (2003). HARM: A Numerical Scheme for General Relativistic Magnetohydrodynamics. *The Astrophysical Journal*, 589(1):444–457. 2, 6

Gittings, M., Weaver, R., Clover, M., Betlach, T., Byrne, N., Coker, R., Dendy, E., Hueckstaedt, R., New, K., Oakes, W. R., Ranta, D., and Stefan, R. (2008). The RAGE radiation-hydrodynamic code. *Computational Science & Discovery*, 1(1):015005. 2

González, M., Audit, E., and Huynh, P. (2007). HERACLES: a three-dimensional radiation hydrodynamics code. *Astronomy and Astrophysics*, 464(2):429–435. 6

Guilet, J., Sato, J., and Foglizzo, T. (2010). The Saturation of SASI by Parasitic Instabilities. *The Astrophysical Journal*, 713:1350–1362. 23

Harten, A., Lax, P. D., and Leer, B. V. (1983). On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws. *SIAM Review*, 25(1):35. 2, 6

Janka, H., Langanke, K., Marek, A., Martinezpinedo, G., and Muller, B. (2007). Theory of core-collapse supernovae. *Physics Reports*, 442(1-6):38–74. 1

Janka, H.-T. (2012). Explosion Mechanisms of Core-Collapse Supernovae. *ArXiv:1206.2503*. 1

Käppeli, R., Whitehouse, S. C., Scheidegger, S., Pen, U.-L., and Liebendörfer, M. (2011). Fish: a Three-Dimensional Parallel Magnetohydrodynamics Code for Astrophysical Applications. *The Astrophysical Journal Supplement Series*, 195(2):20. 21

Khokhlov, A. (1998). Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations. *Journal of Computational Physics*, 143(2):519–543. 2

Kotake, K., Sato, K., and Takahashi, K. (2006). Explosion mechanism, neutrino burst and gravitational wave in core-collapse supernovae. *Reports on Progress in Physics*, 69:971–1143. 1

Kurganov, A., Noelle, S., and Petrova, G. (2001). Semidiscrete Central-Upwind Schemes for Hyperbolic Conservation Laws and Hamilton–Jacobi Equations. *SIAM Journal on Scientific Computing*, 23(3):707. 2, 21

Kurganov, A. and Tadmor, E. (2000). New High-Resolution Central Schemes for Nonlinear Conservation Laws and ConvectionDiffusion Equations. *Journal of Computational Physics*, 160(1):241–282. 2, 4

Landau, L. D. and Lifshitz, E. M. (1959). *Course of Theoretical Physics, Fluid Mechanics*, volume 6. Addison-Wesley, Reading, MA. 11

LeVeque, R. J. (2002). *Finite Volume Methods for Hyperbolic Problems (Cambridge Texts in Applied Mathematics)*. Cambridge University Press. 2, 3, 4, 5, 18

Liebendörfer, M., Messer, O. E. B., Mezzacappa, A., Bruenn, S. W., Cardall, C. Y., and Thielemann, F.-K. (2004). A Finite Difference Representation of Neutrino Radiation Hydrodynamics in Spherically Symmetric General Relativistic Spacetime. *ApJS*, 150:263–316. 27

Linde, T. (2002). A practical, general-purpose, two-state HLL Riemann solver for hyperbolic conservation laws. *International Journal for Numerical Methods in Fluids*, 40(3-4):391–402. 2, 6

Lingerfelt, E., Messer, O., Osborne, J., Budiardja, R., and Mezzacappa, A. (2011). A multitier system for the verification, visualization and management of chimera. *Procedia Computer Science*, 4(0):2076 – 2085. Proceedings of the International Conference on Computational Science, ICCS 2011. 26

Liska, R. and Wendroff, B. (2003). Comparison of Several Difference Schemes on 1D and 2D Test Problems for the Euler Equations. *SIAM Journal on Scientific Computing*, 25(3):995. 16, 19, 20, 21, 25, 41

Londrillo, P. and Del Zanna, L. (2004). On the divergence-free condition in Godunov-type schemes for ideal magnetohydrodynamics: the upwind constrained transport method. *Journal of Computational Physics*, 195(1):17–48. 2, 6

Mezzacappa, A. (2005). Ascertaining the Core Collapse Supernova Mechanism: The State of the Art and the Road Ahead. *Annual Review of Nuclear and Particle Science*, 55(1):467–515. 1

Mignone, a. and Bodo, G. (2005). An HLLC Riemann solver for relativistic flows - I. Hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 364(1):126–136. 2, 6

Mignone, a., Ugliano, M., and Bodo, G. (2009). A five-wave Harten-Lax-van Leer Riemann solver for relativistic magnetohydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 393(4):1141–1156. 2, 6, 23, 25, 27

Miyoshi, T. and Kusano, K. (2005). A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics. *Journal of Computational Physics*, 208(1):315–344. 2, 6, 27

Müller, B., Janka, H.-T., and Dimmelmeier, H. (2010). A New Multi-Dimensional General Relativistic Neutrino Hydrodynamic Code for Core-Collapse Supernovae. I. Method and Code Tests in Spherical Symmetry. *The Astrophysical Journal Supplement Series*, 189(1):104–133. 2

Müller, B., Janka, H.-T., and Marek, A. (2012). A New Multi-Dimensional General Relativistic Neutrino Hydrodynamics Code for Core-Collapse Supernovae II. Relativistic Explosion Models of Core-Collapse Supernovae. *ArXiv e-prints*. 2

Reid, J. (2007). The new features of fortran 2003. *SIGPLAN Fortran Forum*, 26:10–33. 2, 26

Roe, P. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372. 20

Ryu, D., Jones, T. W., and Frank, A. (2000). The Magnetohydrodynamic KelvinHelmholtz Instability: A Threedimensional Study of Nonlinear Evolution. *The Astrophysical Journal*, 545(1):475–493. 25

Sekora, M. D. and Stone, J. M. (2010). A hybrid Godunov method for radiation hydrodynamics. *Journal of Computational Physics*, 229(19):6819–6852. 6

Shu, C. (1997). Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical report, ICASE Report No. 97-65, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center. 2, 4

Shu, C. (1998). Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. *Advanced numerical approximation of nonlinear hyperbolic equations*, (97):325–432. 4, 10

Sod, G. (1978). A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31. 19

Springel, V. (2010). E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *MNRAS*, 401:791–851. 21

Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., and Simon, J. B. (2008). Athena: A New Code for Astrophysical MHD. *The Astrophysical Journal Supplement Series*, 178(1):137–177. 16, 18, 20, 21

Stone, J. M., Hawley, J. F., Evans, C. R., and Norman, M. L. (1992). A Test Suite for Magnetohydrodynamcial Simulations. *The Astrophysical Journal*, 388:415–437. 16

Teyssier, R. (2002). Cosmological hydrodynamics with adaptive mesh refinement. *Astronomy and Astrophysics*, 385(1):337–364. 2

Toro, E. F. (1999). NUMERICA: a Library of Source Codes for Teaching, Research and Applications. `http://www.numeritek.com/`. 20

Toro, E. F. (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer-Verlag, Berlin, Heidelberg. 2, 16, 19, 20

Toro, E. F., Spruce, M., and Speares, W. (1994). Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4(1):25–34. 2, 6, 7

Tsatsin, P., Budiardja, R., Cardall, C., Endeve, E., Marronetti, P., and Mezzacappa, A. (2011). GenASiS: A full GR-RMHD simulation framework: overview, goals, and preliminary tests. In *APS Meeting Abstracts*, page 12006. 27

Woodward, P. and Colella, P. (1984). The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115–173. 21

Woosley, S. and Janka, T. (2005). The physics of core-collapse supernovae. *Nature Physics*, 1(3):147–154. 1

This preprint was prepared with the AAS LaTeX macros v5.2.

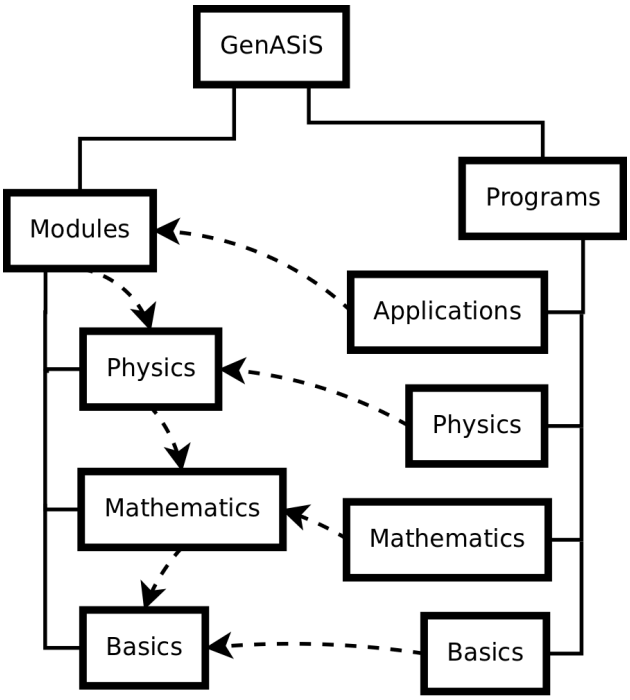

Fig. 1.— High-level structure of the core of GenASiS. Solid lines outline the directory hierarchy, and dashed arrows indicate compilation dependencies.

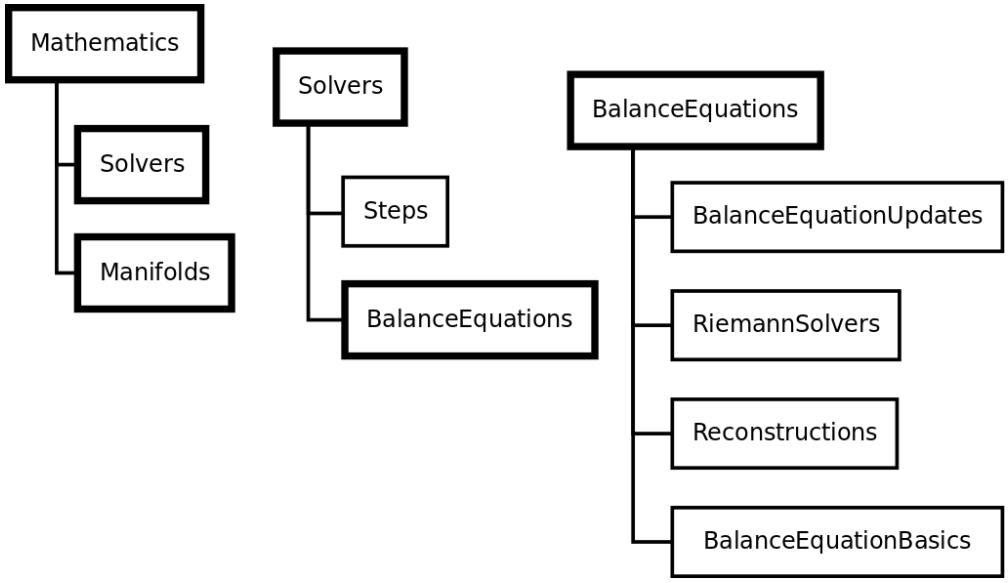

Fig. 2.— *Left:* Structure of `Mathematics`. (For the place of `Mathematics` in the overall scheme of GenASiS, see Figure 1.) *Middle:* Substructure of `Solvers`. *Right:* Substructure of `BalanceEquations`. *All:* Solid lines outline the directory hierarchy. Boxes framed with thinner linewidths denote 'leaf' divisions of the code with no additional subdirectories. The compilation order is from bottom to top; thus dependencies essentially flow in reverse, from top to bottom.

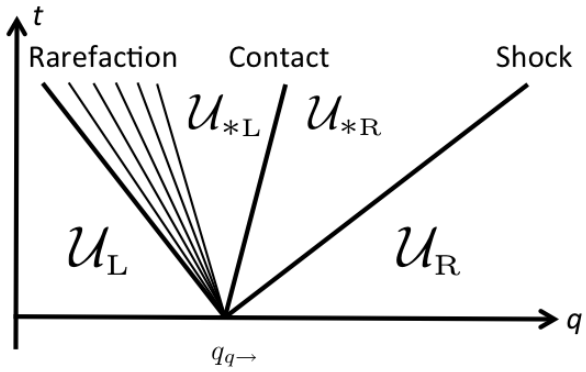

Fig. 3.— Spacetime diagram sketch of a typical Riemann fan for the Euler equations. A discontinuity at $q = q_{q\rightarrow}$ initially ($t = 0$) separates the two constant states $\mathcal{U}_{\mathrm{L}}$ and $\mathcal{U}_{\mathrm{R}}$. For $t > 0$, $\mathcal{U}_{\mathrm{R}}$ and $\mathcal{U}_{*\mathrm{R}}$ are (discontinuously) separated by a right-moving shock wave, $\mathcal{U}_{*\mathrm{R}}$ and $\mathcal{U}_{*\mathrm{L}}$ are separated by a right-moving contact discontinuity, and $\mathcal{U}_{*\mathrm{L}}$ and $\mathcal{U}_{\mathrm{L}}$ are separated by a left-moving rarefaction wave. The rarefaction wave continuously connects $\mathcal{U}_{*\mathrm{L}}$ and $\mathcal{U}_{\mathrm{L}}$. The Riemann solver aims to determine the state of the fluid along $q = q_{q\rightarrow}$ in order to compute the flux of various quantities across the interface in a given time interval (time step).

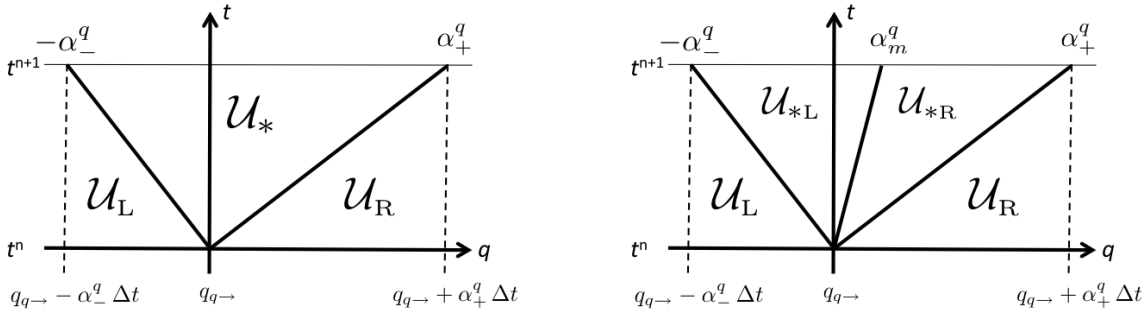

Fig. 4.— Schematic view of Riemann fans adopted for the HLL (left) and HLLC (right) Riemann solvers for hydrodynamics in GenASiS. The HLL solver ignores the entropy wave and assumes a single, constant average state $\mathcal{U}_*$ between the left and right going acoustic waves. The HLLC solver includes the middle (entropy) wave, which separates two constant states $\mathcal{U}_{*\mathrm{L}}$ and $\mathcal{U}_{*\mathrm{R}}$.

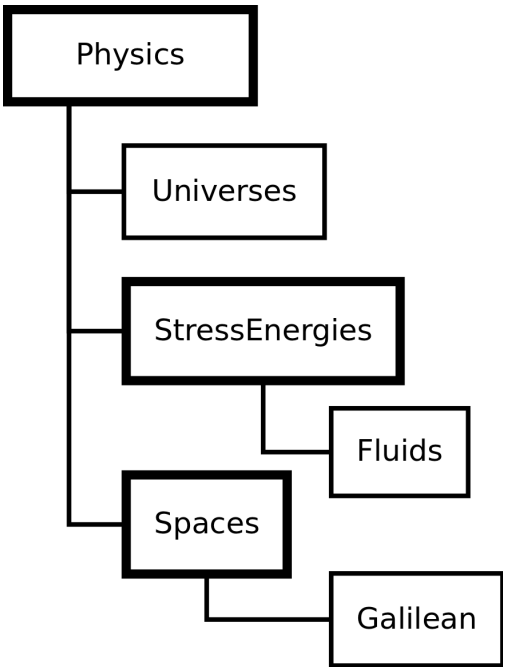

Fig. 5.— Structure of `Physics`. (For the place of `Physics` in the overall scheme of GenASiS, see Figure 1 in Paper I.) Solid lines outline the directory hierarchy. Boxes framed with thinner linewidths denote 'leaf' divisions of the code with no additional subdirectories. The compilation order is from bottom to top; thus dependencies essentially flow in reverse, from top to bottom.

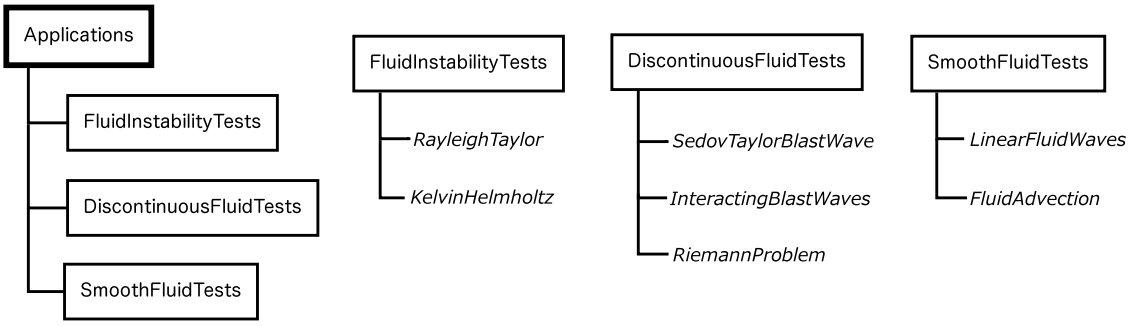

Fig. 6.— *Left:* Structure of `Applications`. (For the place of `Applications` in the overall scheme of GenASiS, see Figure 1.) *Middle Left:* Driver programs in `FluidInstabilityTests`. *Middle Right:* Driver programs in `DiscontinuousFluidTests`. *Right:* Driver programs in `SmoothFluidTests`. *All:* Solid lines outline the directory hierarchy. Boxes framed with thinner linewidths denote 'leaf' divisions of the code with no additional subdirectories. Driver program names are italicized and unboxed.

Table 1: L_1 error norm and convergence rate for 1D advection test with flow Mach number Ma $= 0.6$.

	HLL ($t = 1$)		HLLC ($t = 1$)		HLL ($t = 5$)		HLLC ($t = 5$)	
N_x	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate
16	3.443×10^{-2}	–	2.810×10^{-2}	–	1.306×10^{-1}	–	9.729×10^{-2}	–
32	1.227×10^{-2}	-1.49	1.043×10^{-2}	-1.43	2.831×10^{-2}	-2.21	2.653×10^{-2}	-1.87
64	3.294×10^{-3}	-1.90	2.770×10^{-3}	-1.91	1.046×10^{-2}	-1.44	9.421×10^{-3}	-1.49
128	8.062×10^{-4}	-2.03	6.942×10^{-4}	-2.00	2.936×10^{-3}	-1.83	2.717×10^{-3}	-1.79
256	1.900×10^{-4}	-2.09	1.667×10^{-4}	-2.06	7.600×10^{-4}	-1.95	7.170×10^{-4}	-1.92
512	4.460×10^{-5}	-2.09	4.074×10^{-5}	-2.03	1.900×10^{-4}	-2.00	1.827×10^{-4}	-1.97

Table 2: L_1 error norm and convergence rate for 1D advection test with flow Mach number Ma $= 0.1$.

	HLL ($t = 1$)		HLLC ($t = 1$)		HLL ($t = 5$)		HLLC ($t = 5$)	
N_x	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate
16	1.377×10^{-1}	–	2.777×10^{-2}	–	1.995×10^{-1}	–	9.616×10^{-2}	–
32	2.591×10^{-2}	-2.41	1.026×10^{-2}	-1.44	1.044×10^{-1}	-0.93	2.621×10^{-2}	-1.88
64	8.095×10^{-3}	-1.68	2.709×10^{-3}	-1.92	1.779×10^{-2}	-2.55	9.176×10^{-3}	-1.51
128	1.805×10^{-3}	-2.17	6.758×10^{-4}	-2.00	4.769×10^{-3}	-1.90	2.628×10^{-3}	-1.80
256	3.894×10^{-4}	-2.21	1.616×10^{-4}	-2.06	1.119×10^{-3}	-2.09	6.893×10^{-4}	-1.93
512	8.117×10^{-5}	-2.26	3.804×10^{-5}	-2.09	2.525×10^{-4}	-2.15	1.750×10^{-4}	-1.98

Table 3: L_1 error norm and convergence rate for 2D advection test with flow Mach number Ma $= 0.6$.

	HLL ($t = 1$)		HLLC ($t = 1$)		HLL ($t = 5$)		HLLC ($t = 5$)	
N_x	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate	$L_1(\rho)$	Rate
32	1.019×10^{-2}	–	8.028×10^{-3}	–	3.944×10^{-2}	–	2.722×10^{-2}	–
64	3.577×10^{-3}	-1.51	2.897×10^{-3}	-1.47	8.376×10^{-3}	-2.24	7.892×10^{-3}	-1.79
128	9.485×10^{-4}	-1.92	7.816×10^{-4}	-1.89	3.014×10^{-3}	-1.47	2.698×10^{-3}	-1.55
256	2.311×10^{-4}	-2.04	1.933×10^{-4}	-2.02	8.419×10^{-4}	-1.84	7.715×10^{-4}	-1.81
512	5.421×10^{-5}	-2.09	4.688×10^{-5}	-2.04	2.167×10^{-4}	-1.96	2.026×10^{-4}	-1.93

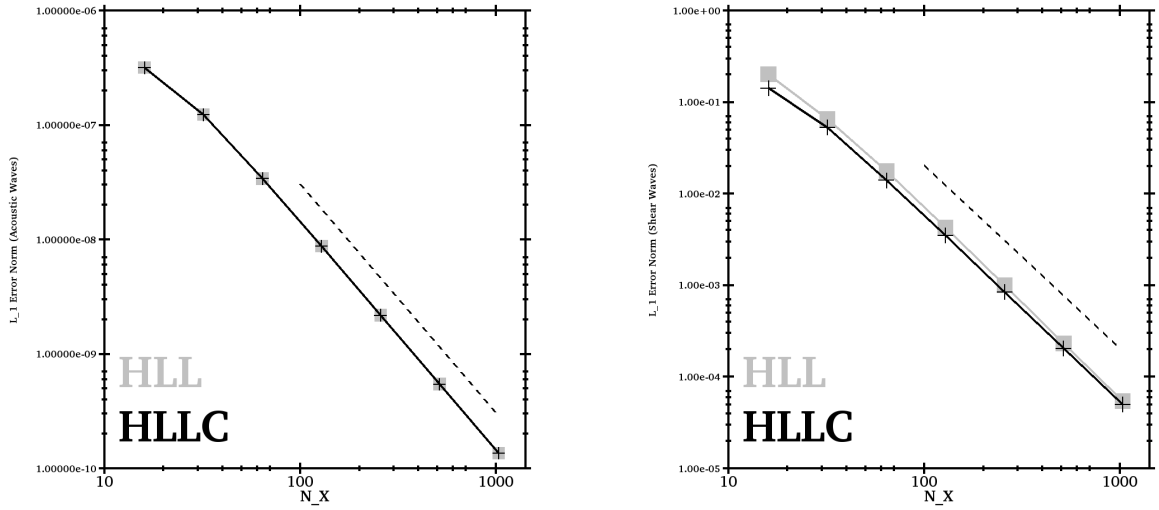

Fig. 7.— Relative L_1 error norms versus number of grid cells N_x from the 1D linear wave tests. The results are obtained with the HLL (grey) and HLLC (black) Riemann solvers in GenASiS. Error norms from the acoustic (left panel) and shear (right panel) wave tests decrease with the expected second-order rate. The dashed reference lines are proportional to N_x^{-2}.

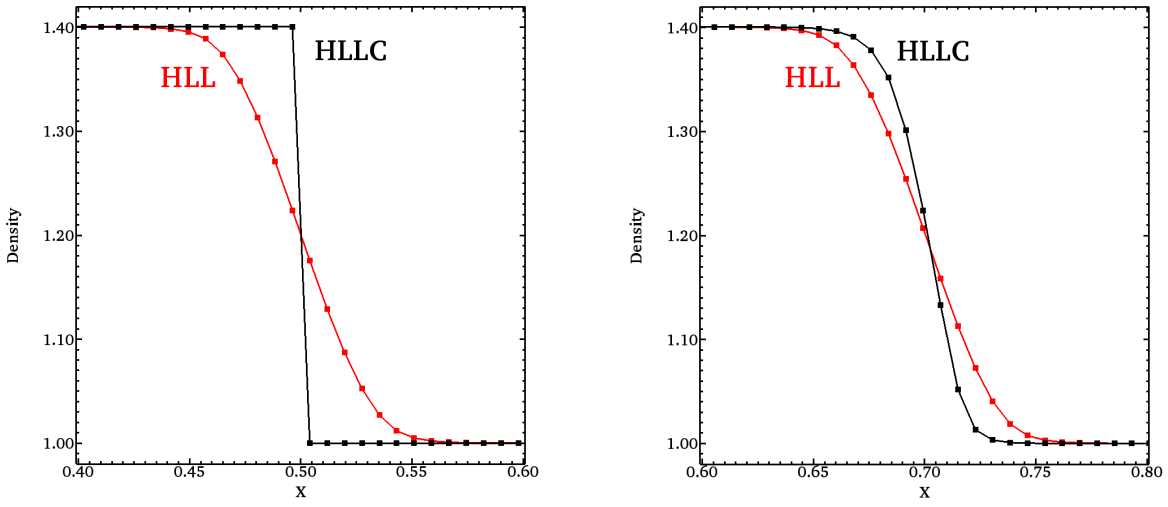

Fig. 8.— Results from the stationary (left) and the slowly moving (right) contact discontinuity at $t = 2$, computed with a grid of 128 zones. The mass density is plotted. Results computed with the HLLC Riemann solver are shown in black, and results computed with the HLL Riemann solver are shown in red.

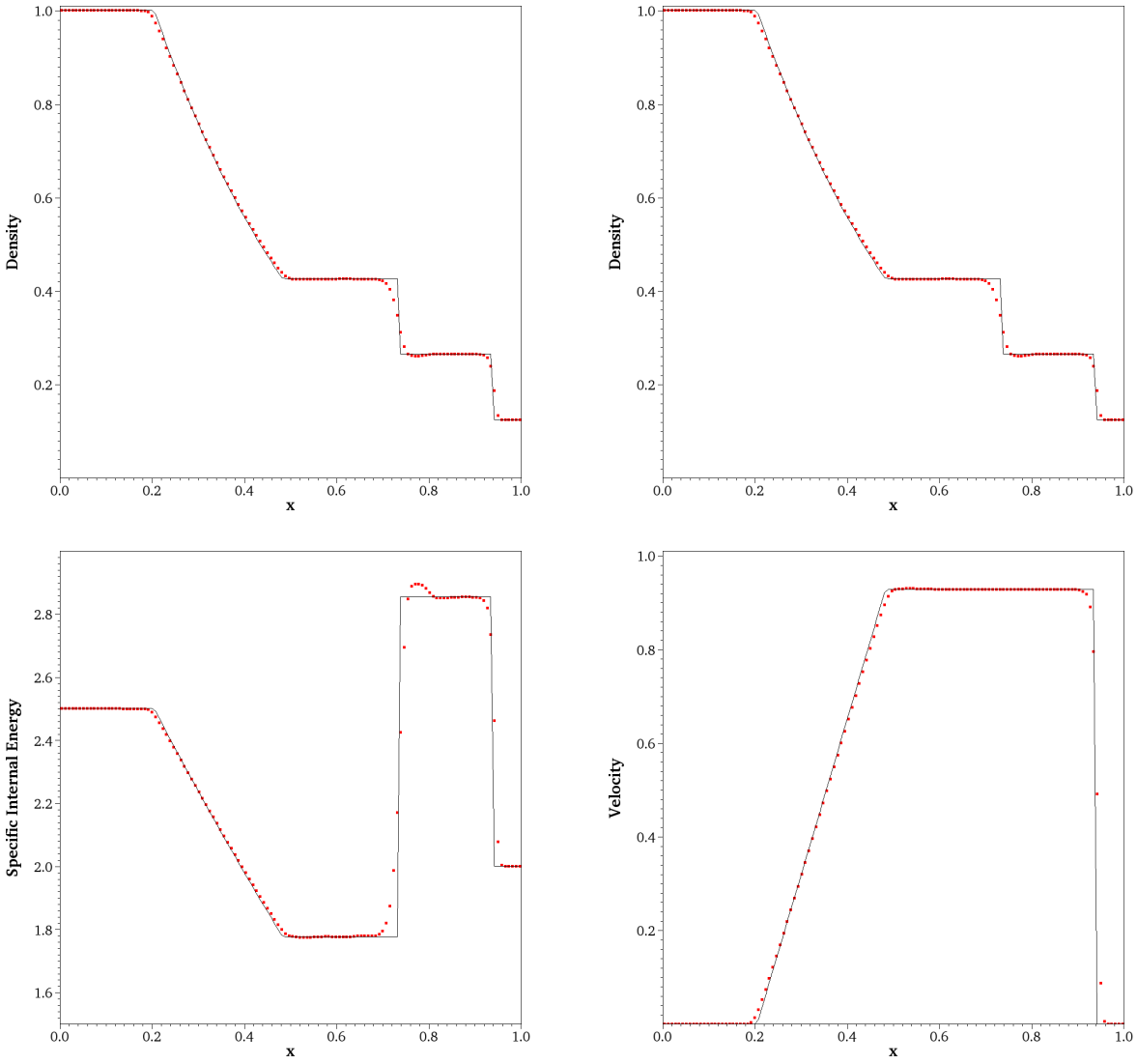

Fig. 9.— Results from running Sod's shocktube test at $t = 0.25$, using 128 cells and the HLLC Riemann solver. The mass density (left) and pressure (right) are plotted in the upper panels, and the specific internal energy (e/ρ; left) and the velocity (v_x; right) are plotted in the lower panels. The analytic solution is shown as a black line in each panel.

Table 4: L_1 error norm and convergence rate for Sod's shocktube problem.

	HLL				HLLC			
N_x	$L_1(\rho)$	Rate	$L_1(p)$	Rate	$L_1(\rho)$	Rate	$L_1(p)$	Rate
32	2.982×10^{-2}	–	2.567×10^{-2}	–	2.809×10^{-2}	–	2.553×10^{-2}	–
64	1.529×10^{-2}	-0.96	1.293×10^{-2}	-0.99	1.452×10^{-2}	-0.95	1.269×10^{-2}	-1.01
128	8.098×10^{-3}	-0.92	6.500×10^{-3}	-0.99	7.803×10^{-3}	-0.90	6.388×10^{-3}	-0.99
256	4.383×10^{-3}	-0.89	3.289×10^{-3}	-0.98	4.256×10^{-3}	-0.88	3.233×10^{-3}	-0.98
512	2.434×10^{-3}	-0.85	1.692×10^{-3}	-0.96	2.362×10^{-3}	-0.85	1.658×10^{-3}	-0.96
1024	1.311×10^{-3}	-0.89	7.911×10^{-4}	-1.10	1.279×10^{-3}	-0.89	7.748×10^{-4}	-1.10
2048	7.495×10^{-4}	-0.81	4.095×10^{-4}	-0.95	7.322×10^{-4}	-0.81	4.013×10^{-4}	-0.95

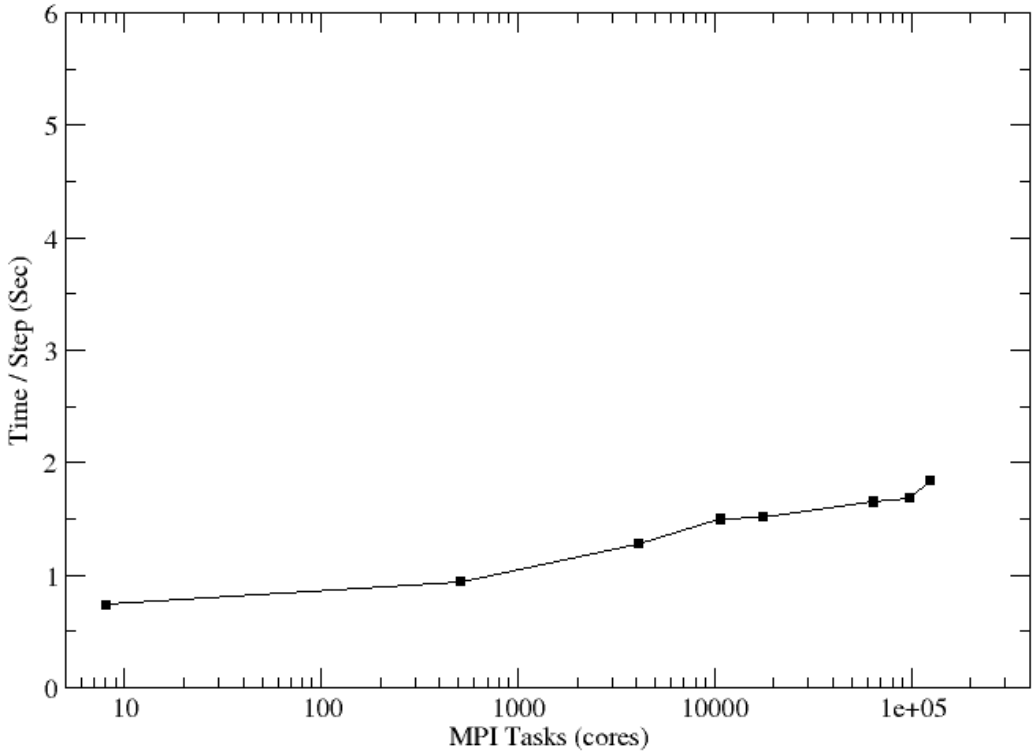

Fig. 10.— MPI weak-scaling of the hydrodynamics algorithms in GenASiS. Sod's shocktube test in 3D was run on a single-level cell-by-cell mesh, keeping the number of computational cells per MPI task fixed to 32^3.

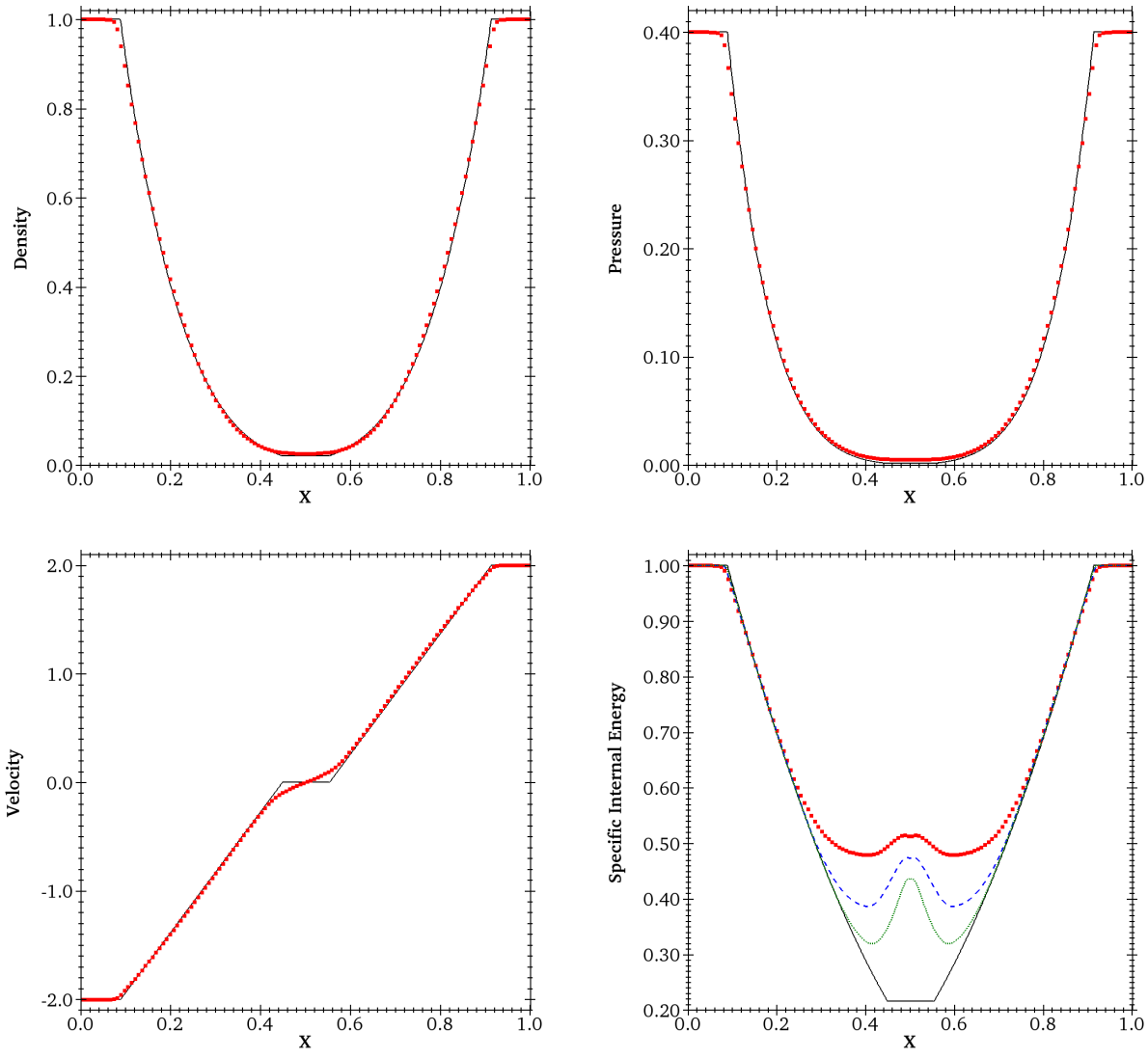

Fig. 11.— Results from the double rarefaction problem at $t = 0.15$, computed with 128 cells and using the HLLC Riemann solver. The density (left) and pressure (right) are plotted in the upper panels, and the velocity (v_x; left) and specific internal energy (e/ρ; right) are plotted in the lower panels. The solid black line is a reference solution obtained with an exact Riemann solver using 1000 cells. For the specific internal energy, results from runs with 256 cells (blue dashed line) and 512 cells (green dotted line) are also plotted.

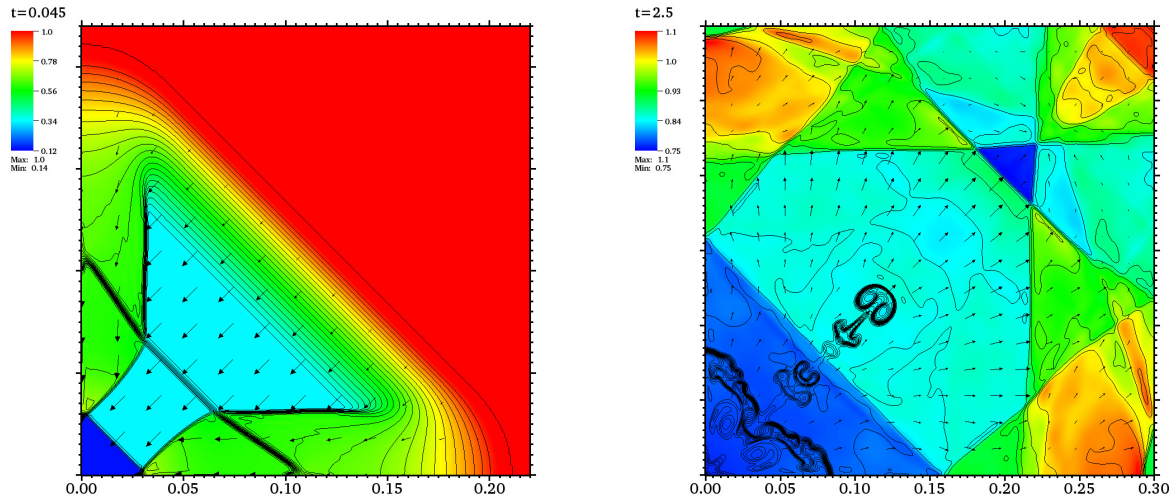

Fig. 12.— Results from running the implosion problem with GenASiS using 400×400 cells and the HLLC Riemann solver. The left and right panels can be compared directly with Figures 4.10 and 4.11 of Liska and Wendroff (2003). The color maps represent the fluid pressure. The density is plotted with contours. Left panel: 36 contours from 0.125 to 1 (in steps of 0.025). Right panel: 31 contours from 0.35 to 1.1 (in steps of 0.025). The arrows indicate the flow velocity.

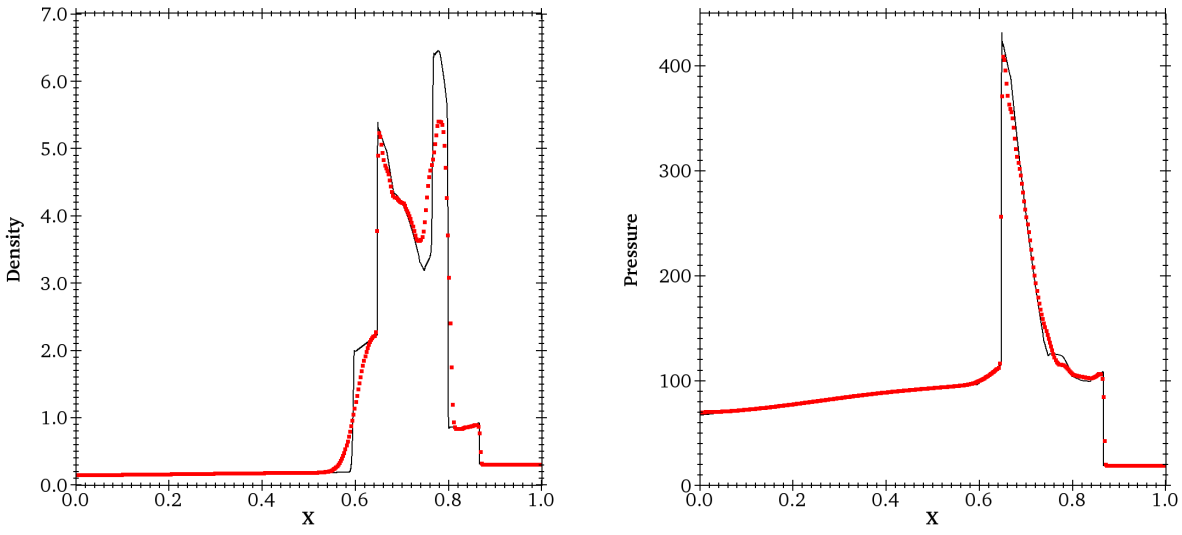

Fig. 13.— Density (left) and pressure (right) from the two-interacting-blast-waves test problem at $t = 0.038$, computed using 400 cells and the HLLC Riemann solver in GenASiS. The solid black line is a high-resolution reference solution obtained by using 10^4 cells.

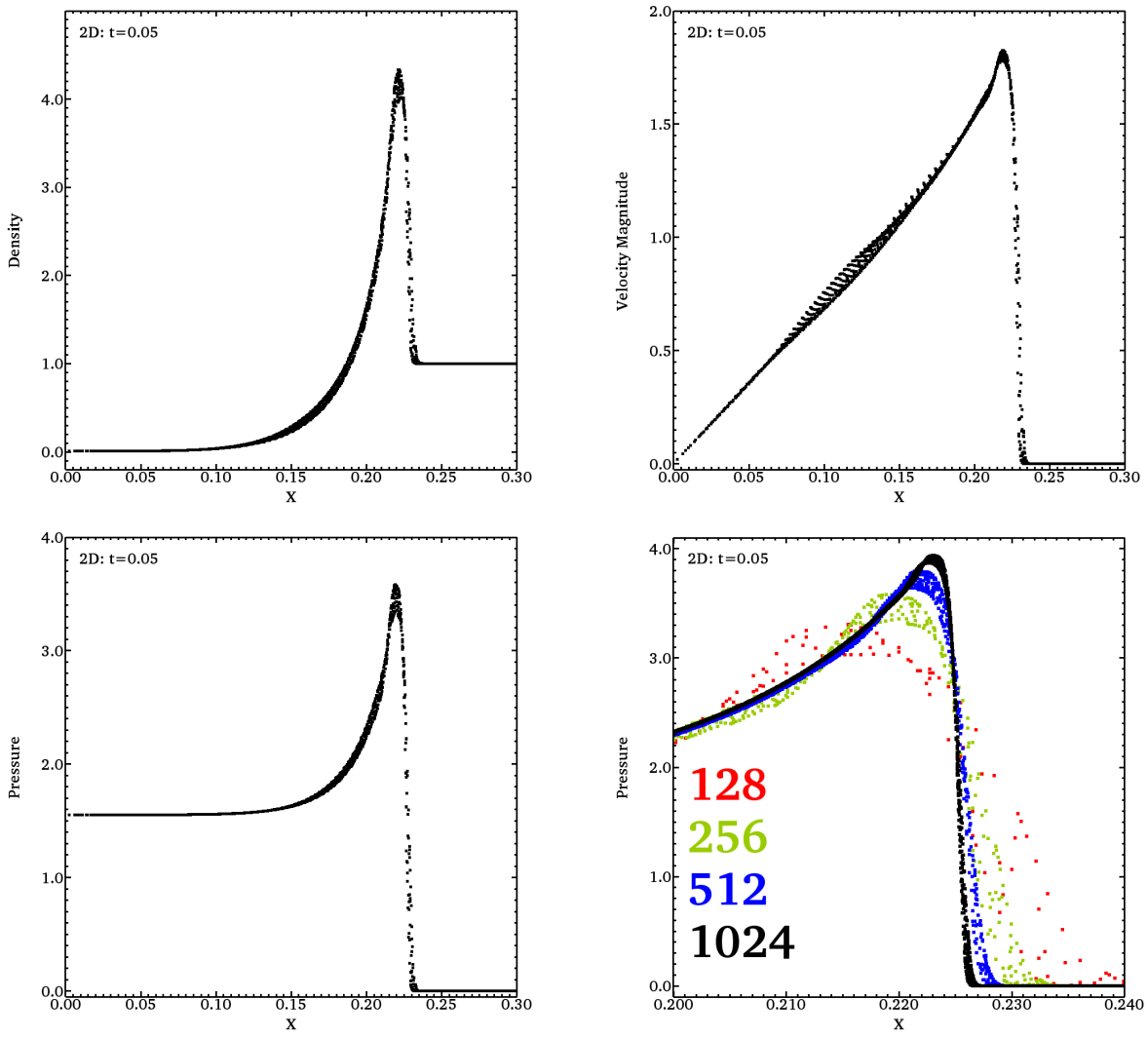

Fig. 14.— Scatter plots of the 2D Sedov-Taylor blast wave problem computed with the HLL Riemann solver.

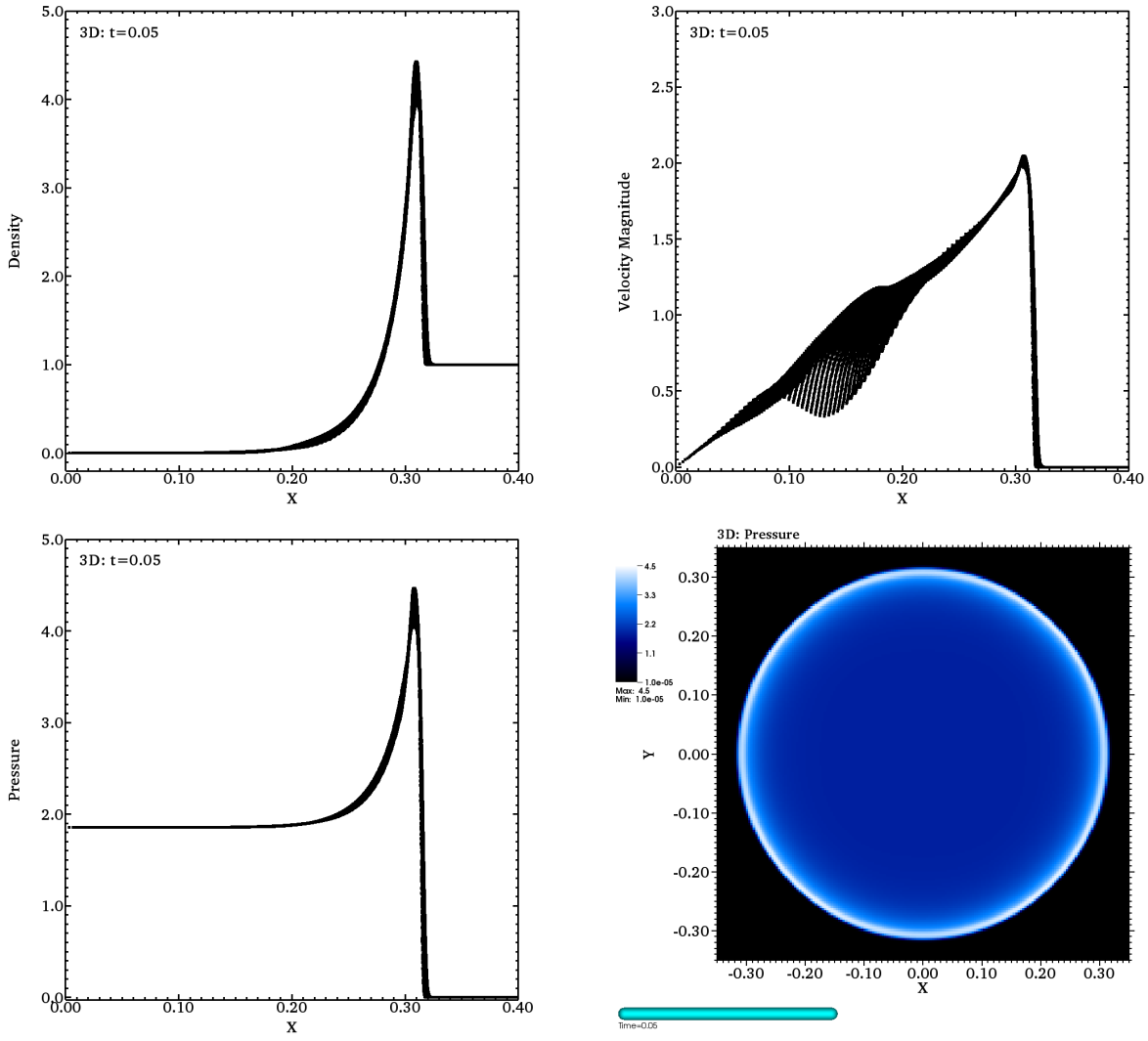

Fig. 15.— Plots of the 3D Sedov-Taylor blast wave problem computed with the HLL Riemann solver with 256^3 zones.

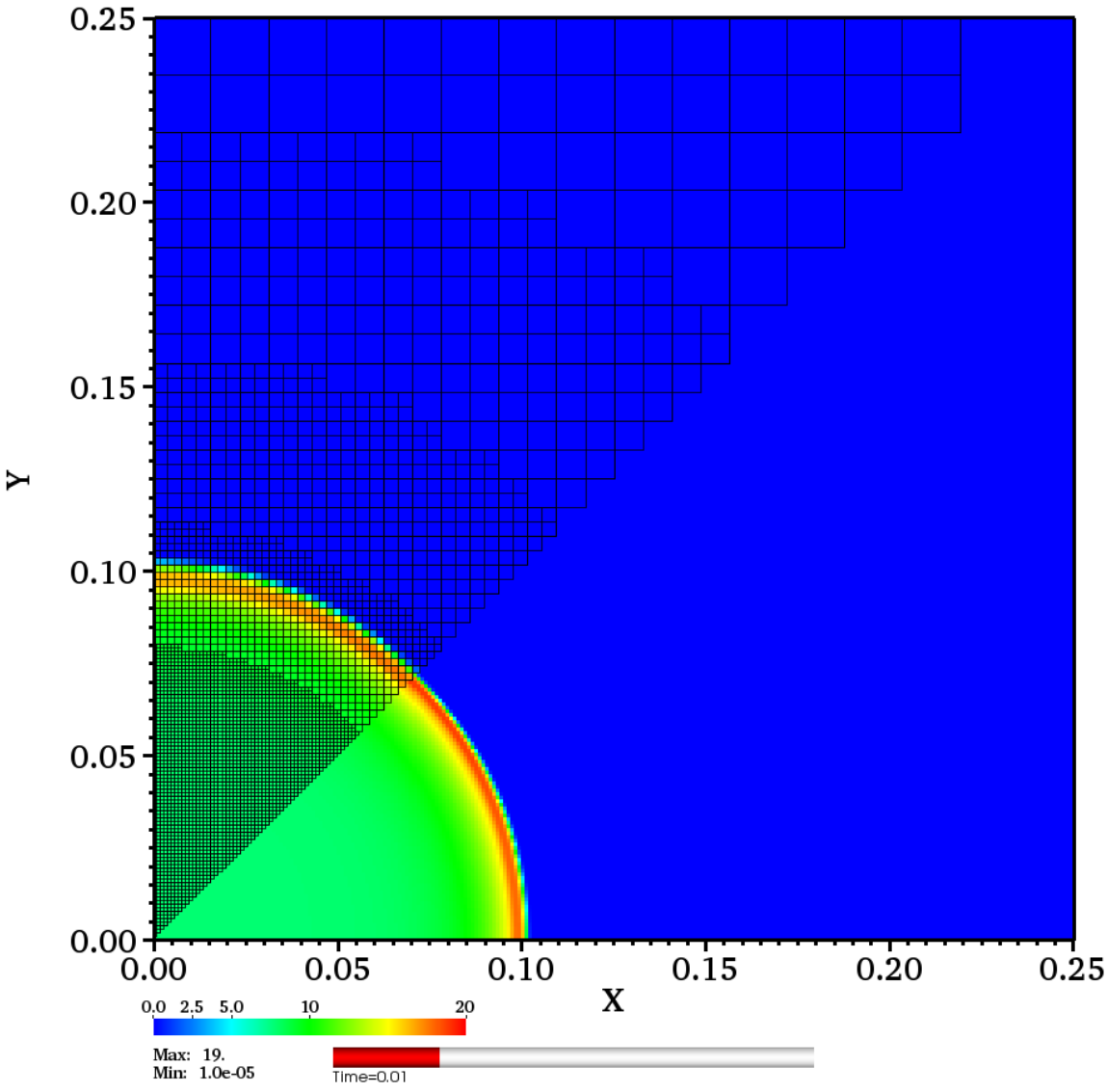

Fig. 16.— Pressure at $t = 0.01$ in the 2D Sedov-Taylor blast wave problem computed with the HLL Riemann solver. Results obtained with the multilevel mesh solver (using seven mesh levels) are shown above the diagonal $y = x$ and compared with single-level results shown below the diagonal. (The multilevel mesh is also shown for $y > x$.)

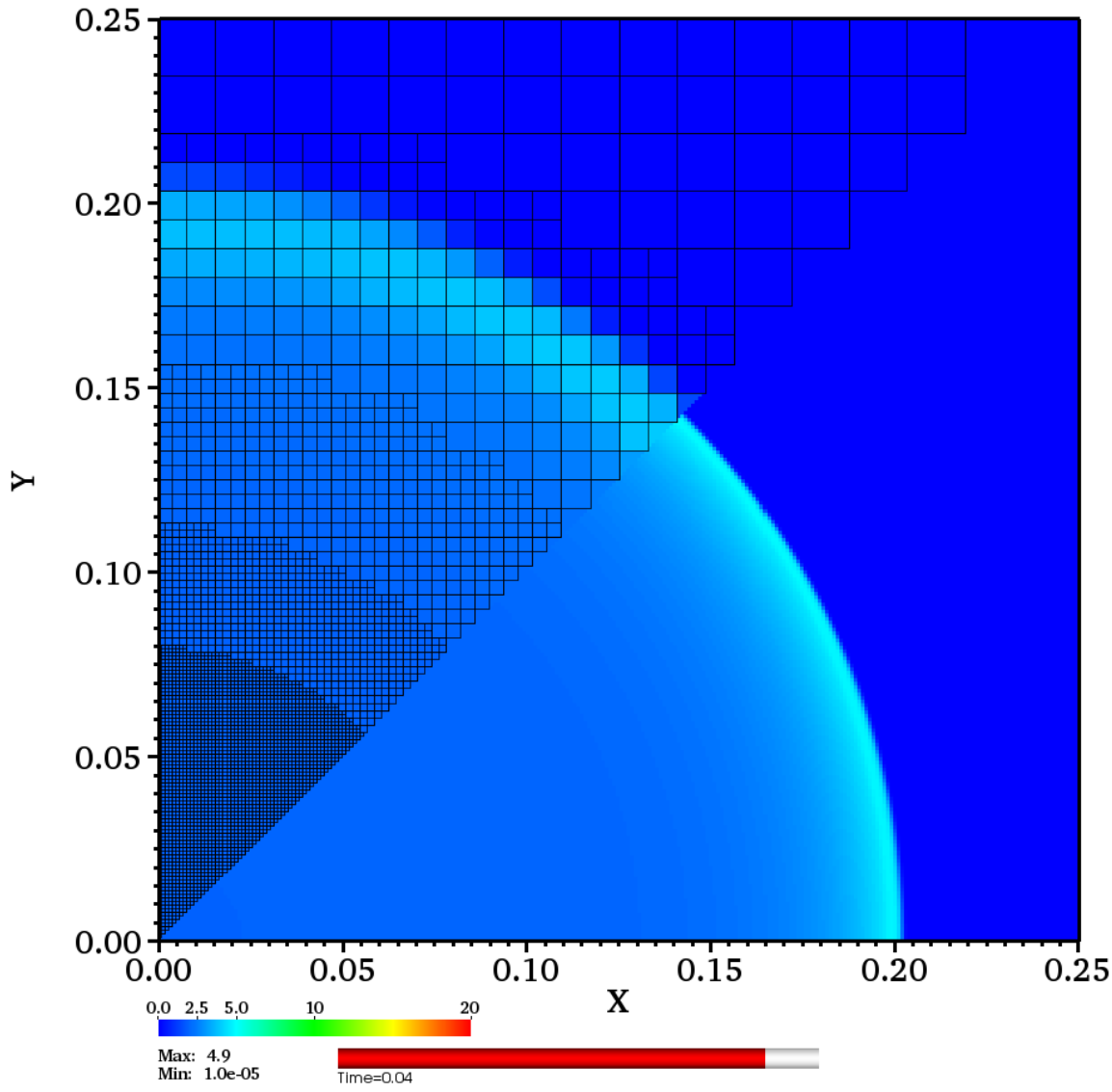

Fig. 17.— Same as Figure 16, but for $t = 0.04$.

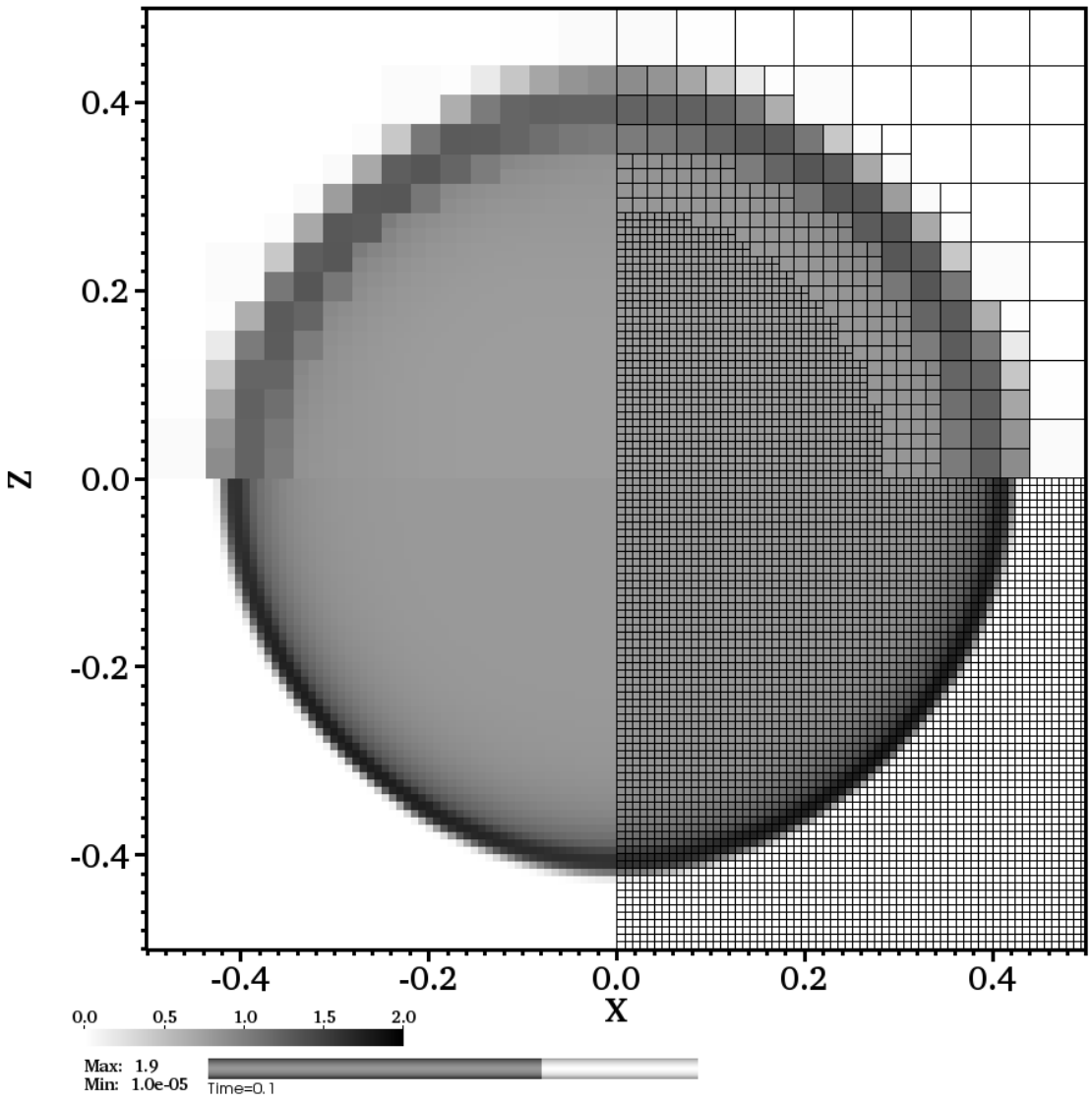

Fig. 18.— Plots of the pressure in the xz-plane ($y = 0$) from the 3D Sedov-Taylor blast wave problem at $t = 0.1$, computed with the HLL Riemann solver. Results obtained with the multilevel mesh solver (using four mesh levels) are shown for $z > 0$, and compared with unigrid results ($z \leq 0$). (The mesh is shown for $x > 0$.)

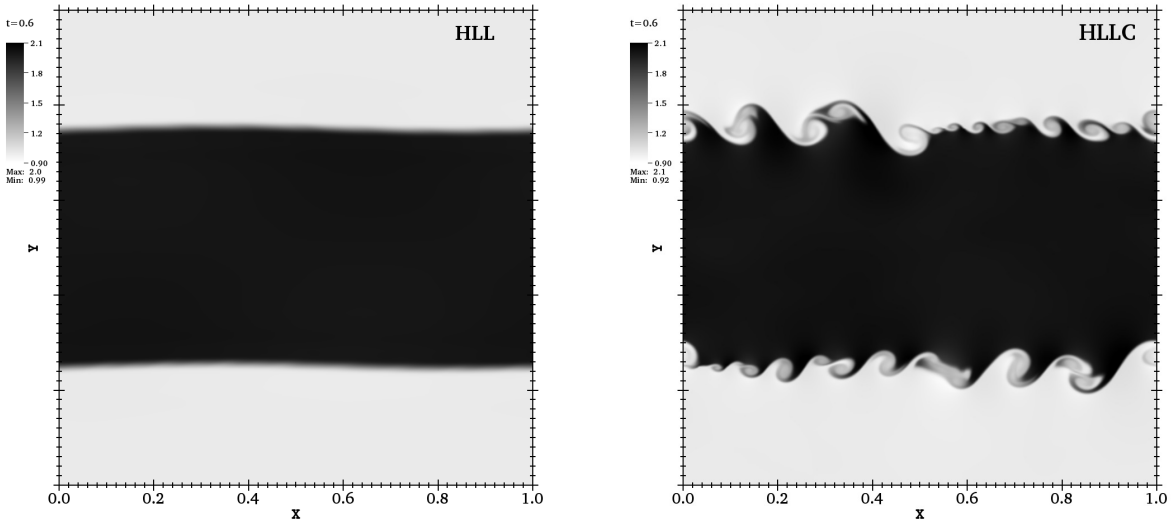

Fig. 19.— Plots of density from the Kelvin-Helmholtz instability test at $t = 0.6$, computed with 512×512 zones. The left and right panels show results computed with the HLL and HLLC Riemann solver, respectively.

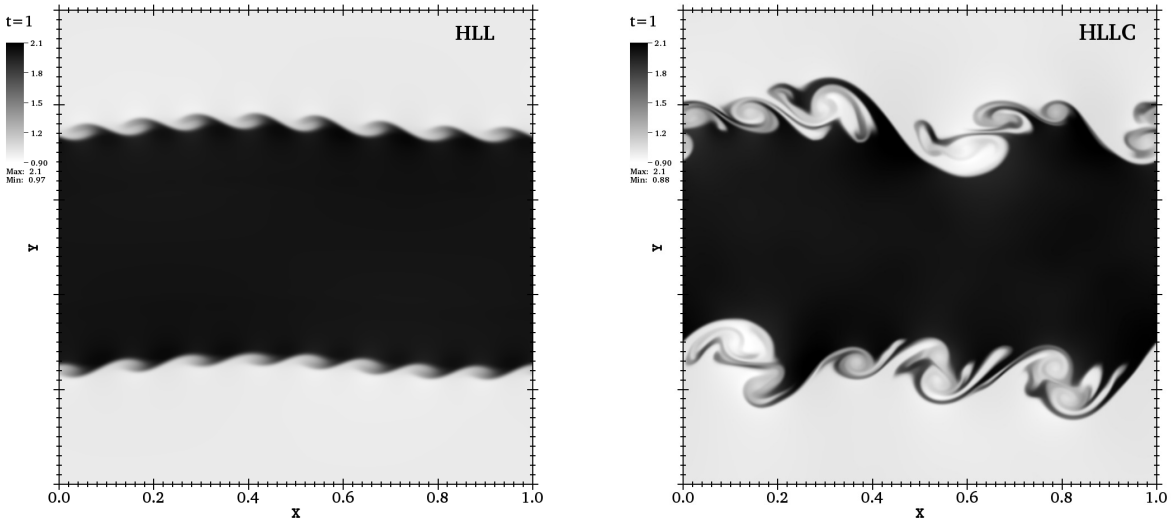

Fig. 20.— Same as Figure 19, but for $t = 1.0$.

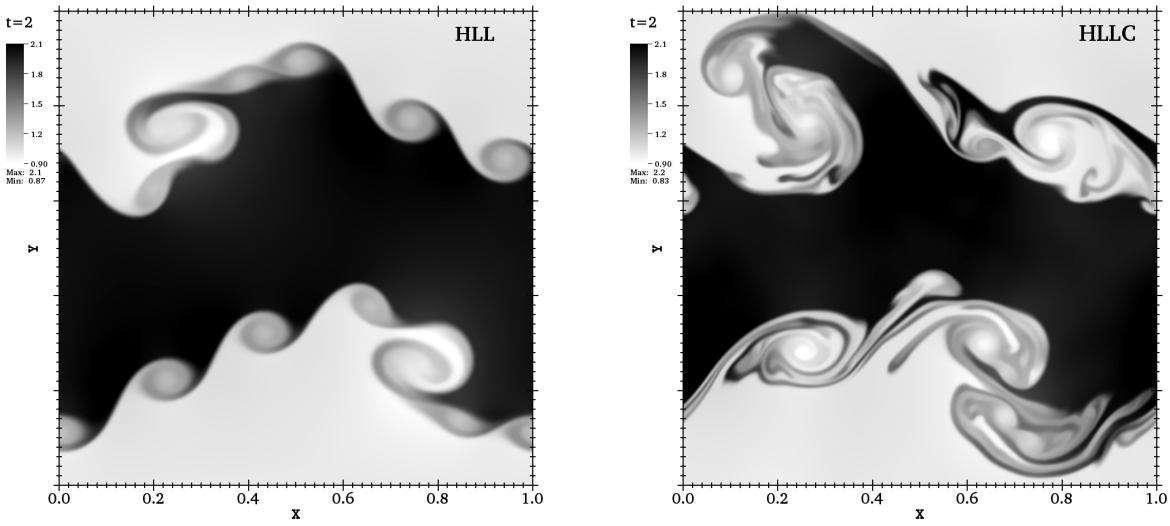

Fig. 21.— Same as Figure 19, but for $t = 2.0$.

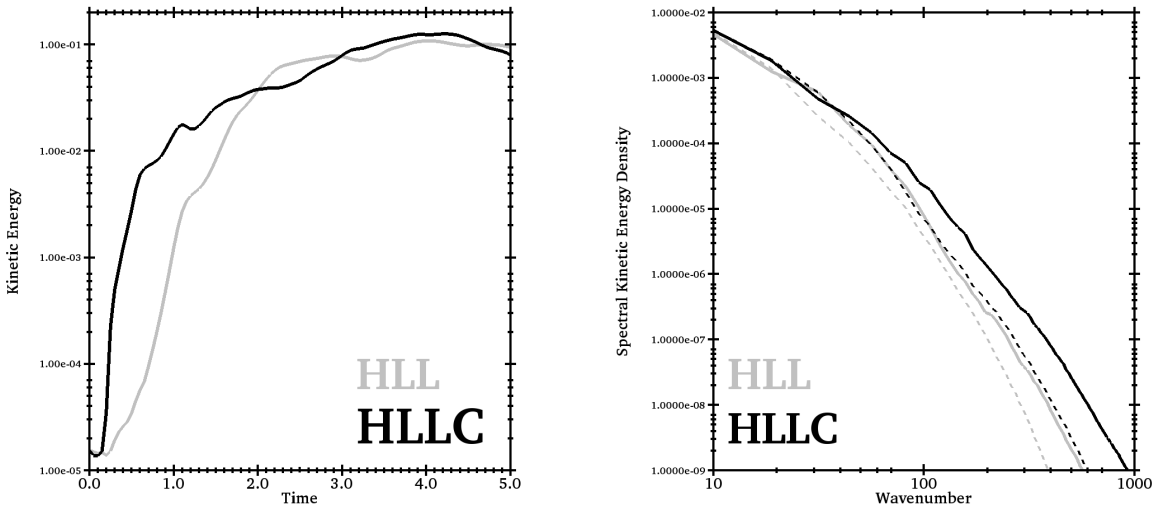

Fig. 22.— Some quantitative results from the Kelvin-Helmholtz instability test: y-component of the kinetic energy versus time (left panel) and the spectral kinetic energy density (right panel). Results obtained with the HLLC and HLL Riemann solvers are represented with black and grey lines, respectively. Solid lines represent results obtained with a 512×512 grid, while dashed lines represent results obtained with a 256×256 grid. The energy spectra in the right panel are time-averaged over the time period extending from $t = 1.9$ to $t = 2.1$.

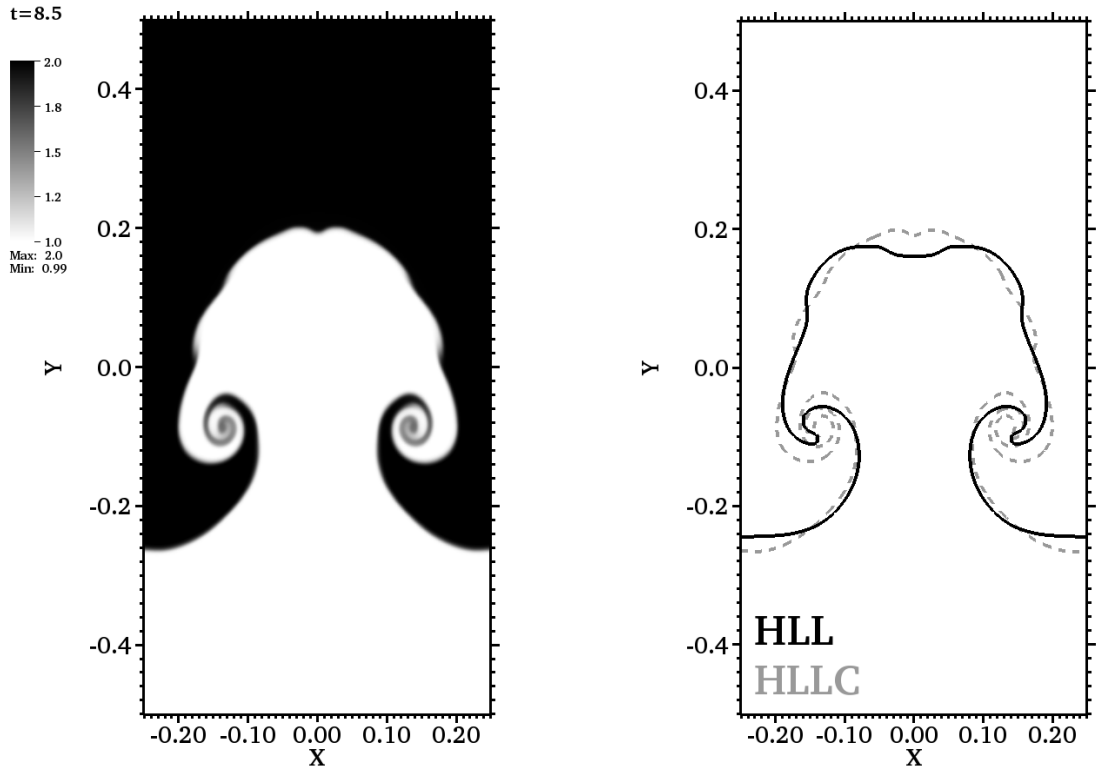

Fig. 23.— Plots of the density from the Rayleigh-Taylor instability test at $t = 8.5$, computed with 256×768 zones. The left panel shows results obtained the HLLC Riemann solver. In the right panel we plot contours of constant density $\rho = 1.25$: computations using the HLL (solid black) and HLLC (dashed grey) Riemann solvers are compared.